

A Thesis Submitted for the Degree of PhD at the University of Warwick

Permanent WRAP URL:

<http://wrap.warwick.ac.uk/135026>

Copyright and reuse:

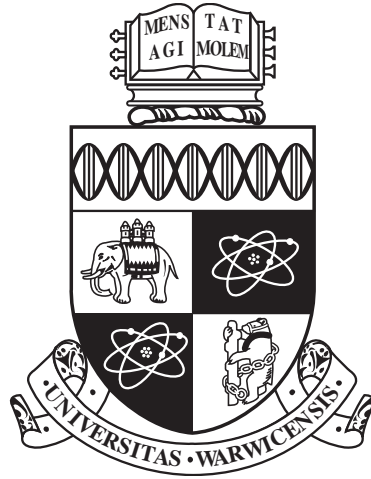
This thesis is made available online and is protected by original copyright.

Please scroll down to view the document itself.

Please refer to the repository record for this item for information to help you to cite it.

Our policy information is available from the repository home page.

For more information, please contact the WRAP Team at: wrap@warwick.ac.uk



**Future state maximisation as an intrinsic
motivation for decision making**

by

Henry Charlesworth

Thesis

Submitted to the University of Warwick

for the degree of

Doctor of Philosophy

Mathematics of Systems

July 2019

THE UNIVERSITY OF
WARWICK

Contents

Acknowledgments	iii
Declarations	iv
Abstract	v
Chapter 1 Introduction	1
Chapter 2 Background and related Work	5
2.1 Intrinsic motivation	6
2.2 The “empowerment” framework	8
2.2.1 Formalism	9
2.2.2 Previous work	11
2.3 The “causal entropic forces” framework	13
2.4 Collective motion	15
Chapter 3 Quantifying future state maximisation	20
3.1 Discrete path/final state counting with adjacency matrices	20
3.1.1 Maze solving application	26
3.1.2 Comparison of the two approaches	29
3.1.3 Extension to discrete Markov chains	30
3.2 Future state difference comparison	32
3.2.1 Application to a toy maze	35
3.2.2 Application to a simple game	36
Chapter 4 Intrinsically motivated collective motion	42
4.1 Description of the base model	42
4.2 Simulations of the base model: collective motion	45

4.3	Alternative models for the predicted trajectories of other agents in the swarm	51
4.4	Training a neural network to mimic the FSM procedure	55
4.5	The importance of collision detection	59
4.6	Other variations in the model	60
4.6.1	Elliptical agents	60
4.6.2	Blind spot in the visual field of view	62
4.7	A more continuous measure of the degeneracy of visual states	64
4.7.1	More continuous action selection	68
4.8	Simulation of larger swarms	69
4.9	Controlling the swarm	72
4.9.1	Controlling the swarm trajectory	74
4.9.2	Guiding the swarm to follow an arbitrary trajectory	77
4.9.3	Future state maximisation at different scales	78
Chapter 5	Threat avoidance by active particles	83
5.1	Description of the model	84
5.1.1	Derivation of a fitness functional	85
5.2	Solving for the probability distribution in free space	87
5.3	Solving the first passage problem	88
5.4	Approximate analytical solution when diffusion is dominant	94
5.5	Full numerical solution for the optimal drift velocity	97
5.5.1	Performance of numerical solution	106
5.6	Results	108
Chapter 6	Conclusions and outlook	112
6.1	Ideas for future work	114
Appendix A	List of supplementary movies	119

Acknowledgments

First and foremost I have to thank my supervisor Matthew Turner. I have enjoyed working with him immensely and I am extremely grateful for all of his guidance, support and enthusiasm throughout the entire duration of my time here.

I would also like to thank all of the students and staff in the Centre for Complexity Science for creating such an enjoyable and stimulating environment to be work in.

Finally, I would like to thank my parents for their unconditional support in everything that I do!

Declarations

This thesis is submitted to the University of Warwick in support of my application for the degree of Doctor of Philosophy. The work presented here is my own, and has not been submitted in any previous application for any degree or professional qualification.

Parts of Chapter 4 have been published in *Proceedings of the National Academy of Sciences* (<https://doi.org/10.1073/pnas.1822069116>, July 17 2019), and parts of Chapters 3,4 and 5 are also currently being prepared for submission to peer review.

Abstract

The concept of an “intrinsic motivation” is used in the psychology literature to distinguish between behaviour which is motivated by the expectation of an immediate, quantifiable reward (“extrinsic motivation”) and behaviour which arises because it is inherently useful, interesting or enjoyable. Examples of the latter can include curiosity driven behaviour such as exploration and the accumulation of knowledge, as well as developing skills that might not be immediately useful but that have the potential to be re-used in a variety of different future situations. In this thesis, we examine a candidate for an intrinsic motivation with wide-ranging applicability which we refer to as “future state maximisation”. Loosely speaking this is the idea that, taking everything else to be equal, decisions should be made so as to maximally keep one’s options open, or to give the maximal amount of control over what one can potentially do in the future. Our goal is to study how this principle can be applied in a quantitative manner, as well as identifying examples of systems where doing so could be useful in either explaining or generating behaviour.

We consider a number of examples, however our primary application is to a model of collective motion in which we consider a group of agents equipped with simple visual sensors, moving around in two dimensions. In this model, agents aim to make decisions about how to move so as to maximise the amount of control they have over the potential visual states that they can access in the future. We find that with each agent following this simple, low-level motivational principle a swarm spontaneously emerges in which the agents exhibit rich collective behaviour, remaining cohesive and highly-aligned. Remarkably, the emergent swarm also shares a number of features which are observed in real flocks of starlings, including scale free correlations and marginal opacity. We go on to explore how the model can be developed to allow us to manipulate and control the swarm, as well as looking at heuristics which are able to mimic future state maximisation whilst requiring significantly less computation, and so which could plausibly operate under animal cognition.

Chapter 1

Introduction

In recent years the quest to develop “artificial intelligence” has made some remarkable progress. Computers are now at the stage where they can achieve super-human performance at tasks varying from image classification[1] to cancer detection[2]. They have also been trained to master the game of Go[3], comfortably beating the world’s best players; something previously thought to be decades away. However, whilst these triumphs should not be downplayed, it is nevertheless true that current techniques are only able to produce systems which are competent in the extremely narrow range of tasks for which they have been trained to solve. They do not generalise well to behaving sensibly, or “intelligently”, in other situations; for example the neural network which was trained to play Go is not also able to classify images, or play Chess, or indeed do anything other than play Go. The goal of creating what is sometimes referred to as “artificial general intelligence”, i.e. machines which are competent over a broad range of challenging tasks in the way that a human is has made significantly less progress. One possible explanation for this is that we still don’t have a particularly good understanding of what intelligence actually is - we know it when we see it, however actually pinning down exactly what it is that makes something “intelligent” is hard to do. In particular, we don’t have a full set of simple, task-independent principles which we could say an “intelligent” agent should be expected to follow in any given situation that it finds itself in.

Although we do not claim to provide a definitive set of such principles in this thesis, we do look at one potential candidate which could provide a motivation for sophisticated behaviour in a wide range of different scenarios, and which might be associated in some sense with intelligence. We refer to this as the principle of “future state maximisation” (FSM). Fundamentally, this principle states that taking everything else to be equal it is

best to keep one’s options open as much as possible, i.e. to have access to the largest variety of potential future states. The motivation here is that in an uncertain world an agent which follows this principle would be preparing itself to cope with the widest range of potential scenarios that it might encounter at a later point — and hence would in some sense be maximising the amount of control it has over its future environment. Aiming to achieve mastery over its environment should naturally lead to things such as the development of re-usable skills as well as the desire to learn more about how its environment works, since both of these will generally unlock access to more possibilities in the future. The amount of control an agent has over its environment could also be seen as a useful proxy for evolutionary fitness in many situations. Whilst clearly the ultimate goal of an organism is to survive and reproduce, this direct feedback is fairly sparse and occurs over a long time, at least relative to the organism’s lifetime. Over shorter time-scales, it is difficult to think of a situation where an organism that attains more control over its environment would become less likely to survive, and easy to think of ways in which this could benefit the organism — for example, activities that lead to the efficient gathering of energy and useful resources would naturally fit into this framework. As such, we might expect that behaviours and heuristics which make it seem as if the organism is trying to maximise its number of future possibilities could arise quite naturally. This could make FSM a useful principle in understanding a whole range of different animal behaviours, in addition to its potential use in generating decisions for “intelligent matter” (where constituent “particles” are able to carry out calculations and make choices about what to do) and more general artificial intelligence systems as well. Indeed, the primary aim of this thesis is to analyse different ways in which this principle can be explicitly applied in a quantitative manner, as well as identifying situations where it could be useful for generating behaviour or understanding why organisms make the particular kind of decisions that we observe.

In Chapter 2 we discuss in detail how the principle of future state maximisation can be linked to the idea of *intrinsic motivations* for behaviour[4]; a concept originating in the psychology literature. An “intrinsic motivation” refers to anything which provides an agent with an incentive to act in a situation where there is no expectation of an immediate or clearly defined external reward to be received (e.g. discovering food). This includes activities such as playing, curiosity-driven exploration and the accumulation of knowledge; none of which provide an immediately obvious benefit to an organism’s survival. We then give a brief overview of two previous attempts that have tried to formalise ideas related to FSM quantitatively; the *empowerment* framework[5], which

takes an approach based on information theory, and the *causal entropic forces* framework which instead uses the language of thermodynamics. We discuss the similarities and the differences between these two approaches as well as their relative benefits and the problems that they both face. These include difficulties in applying them to continuous systems, as well as a large computational expense for applying them to all but the simplest toy systems.

In Chapter 3 we look at some different approaches that can be used to quantify FSM. Here, we start by looking at how we can explicitly count the number of future states or future paths available to any simple discrete system which can be represented on a graph. We find that in this restrictive special case this can be analysed very efficiently simply by taking powers of the adjacency matrix, allowing us to model future trajectories far ahead into the future. We use these to study the properties of some toy systems as well as then looking at an application to maze solving. After this, we introduce a new measure based on a similar philosophy to the established frameworks, but which can be applied more straightforwardly to systems with a continuous state/action space. We apply this to a simple game, demonstrating how we can maximise future options over a much longer time horizon by supplementing a shorter FSM procedure with a longer random search.

Chapter 4 describes our primary application of the principle of future state maximisation. Here we introduce a group of motile agents equipped with simple visual sensors and that move so as to *maximise the number of possible visual states that they have access to in the future*. When each agent individually applies this procedure, we show that a highly aligned and coordinated swarm with a well regulated density spontaneously emerges under a wide range of conditions. We discuss how some of the properties that arise from this model are strikingly similar to those observed in data on murmurations of starlings. Next, we show how it is possible to train a neural network which is able to qualitatively mimic the results of this model very closely whilst only having access to currently available visual information — significantly reducing the amount of computation required. This demonstrates that it is possible to generate a heuristic closely mimicking the full future state maximisation procedure, but which is simple enough that it could be biologically accessible. As we look at swarms made up of a larger number of agents we find that the dynamics which emerge are remarkably rich, providing an example of the type of complex behaviour that can be generated under this simple principle. For an agent in the swarm to apply the principle of FSM we shall see that it is

necessary for it to have a predictive model for how it expects the other agents to move. What the agents expect the other agents to do can have a significant impact on what they will *actually* do when they apply FSM, and we explore the effect of using a number of different predictive models for the other agents' behaviour. We then go on to show how using a particular choice for this predictive model can allow us to manipulate the swarm to follow any macroscopic trajectory of our choosing. This is used as the basis for a model that looks at applying the FSM principle at two different scales — both at the level of the individual agents as well as at the level of the swarm's centre of mass trajectory.

Finally in Chapter 5 we study a fully continuous decision making process in terms of time, state space and action space. The system considered is a one-dimensional Brownian particle which is capable of generating a time-dependent drift velocity, in the presence of a stationary threat. Rather than directly applying the principle of future state maximisation, here we set up a first-passage problem to calculate the probability of the particle dying under an arbitrary drift velocity and then derive a fitness functional based on this for the particle to optimise over some future time horizon. We argue that this fitness is somewhat analogous to the number of future paths which the particle has available to it over its expected future lifetime, such that loosely this is still an application of FSM. Doing this, we are able to obtain an approximate analytical solution as well as a full numerical solution for the optimal drift velocity. Again, we find that a relatively simple heuristic emerges which closely mimics the exact results of an exhaustive calculation.

Chapter 2

Background and related Work

Consider the following simple question: taking everything else to be equal, is it preferable to own a car or not? If we assume that one knows how to drive the car and ignore practical considerations such as running costs and potential environmental impacts then it seems reasonable to assume that most people would agree that it is preferable — but why exactly is this? The act of acquiring a car does not satisfy any simple “animalistic” drives such as thirst or hunger, and whilst, for some, driving may be an inherently enjoyable experience this is not true in general. However, by owning a car one is able to get to places far more quickly than would otherwise be possible, freeing up time for other activities. Additionally one is also able to get to many other places which previously would have taken an impractically long time to reach by foot or public transport (and so were, in effect, inaccessible). In essence, what owning a car does is to increase the number of future possibilities that are available to an owner. We can easily think of other cases like this; for example consider the question of why people go to university. Whilst sometimes it may be because they have a very specific goal in mind, more often the reason is that obtaining a degree will give them access to a much wider range of job opportunities than they would have otherwise. Simpler examples include questions such as why it is preferable to be rich rather than poor, or to be healthy rather than sick. In both cases it is clear that the former provides one with a far greater freedom of choice as well the ability to deal with a variety of unseen problems which might emerge in the future.

These examples demonstrate the potentially wide reach that future state maximisation could have in understanding a range of different decision making processes. Whilst generally individuals don’t tend to think about their choices explicitly in terms of how

they will affect a number of future possibilities, often their intuitions about what would constitute an intelligent decision matches up with what FSM would suggest. The principle is simple and universal, in the sense that it can be applied to any agent (or group of agents) occupying any environment, and yet, despite this, relatively little work has been done in applying this principle in a quantitative manner. In this chapter we start by making a connection between the principle of future state maximisation and the idea of “intrinsic motivations” — studied widely in the psychology literature. We then give a summary of two previous approaches that actually have attempted to formalise something similar to FSM in a quantitative way; the empowerment framework[5] and the causal entropic forces framework[6]. Finally, since the primary application of FSM in this thesis is to building a model of collective motion, we give a brief overview of some of the previous work which has tried to understand and model different types of collective motion seen in nature.

2.1 Intrinsic motivation

The concept of an intrinsic motivation[4][7] is used in psychology to explain the incentive for certain types of behaviour which do not directly lead to (or have the expectation of) an obvious external reward for the decision making agent. Examples include activities such as the desire to explore novel environments, the desire to learn and accumulate knowledge and the desire to acquire skills which can be applied to a variety of different situations. In each of these cases there is no clear gain made by the agent in terms of what is crucial for its immediate survival, yet nevertheless the behaviour is somehow intrinsically useful or enjoyable such that the reward is inherent in the activity itself. More recently this idea has started to gain traction within the reinforcement learning literature[7, 8, 9, 10], where rewards for behaviour of this kind are incorporated on top of the “extrinsic rewards” which are explicitly related to the final objectives being sought.

Initially the term was introduced in response to a previous theory of motivation proposed by Hull[11] which argued that all primary motivations could be explained in terms of *drives*, i.e. temporary physiological deficits that need to be regulated. The idea is based on the concept of homeostasis, i.e. the ability of the human body to maintain internal stability despite changes in its external environment (the most obvious example being the regulation of body temperature). In this theory, a drive refers to a state of tension which is unpleasant for the organism and that it wishes to reduce so as to maintain a kind of internal, homeostatic equilibrium. Simple examples include the need for food

and the need for warmth, which can be satisfied by the organism finding food to eat or sitting by a fire respectively. Hull postulated that all behaviour was motivated by trying to reduce one of an organism’s many drives, and that the organism would learn to repeat any behaviours that were regularly able to reduce these. In other words the theory proposes that all behaviour is generated so as to try and counteract disturbances to a “normal” equilibrium condition of the organism. Although drive reduction theory seems like a natural way to explain things such as hunger and thirst, under this view things such as curiosity would derive their motivational potency only via secondary reinforcement, i.e. the organisms would learn that these behaviours can help to satisfy their primary drives (this is analogous to classical conditioning[12]). This view was cast into doubt by various experiments, including a famous study carried out by Harlow[13] where it was reported that rhesus monkeys would spend hours solving complicated mechanical puzzles without receiving any rewards at all (this was also the study in which the term “intrinsic motivation” was first coined). Whilst attempts were made to reconcile these experiments with Hull’s theory by postulating new drives such as a drive to explore[14], this was ultimately unsuccessful as there would not appear to be any homeostatic function for such a drive, nor can it be related in any obvious way to any internal deficit within the organism. Furthermore, Klopff[15] argued that it really makes more sense to view activities such as exploration and knowledge-gathering in terms of an optimisation process rather than equilibrium-seeking process — since these are activities which allow for open-ended improvement.

This led White[16] to argue that it was better to emphasize how drives for activities such as exploration, manipulation and knowledge-gathering differ from the more conventionally biologically relevant, homeostatic drives. He proposed to bring them under the heading “competence”, arguing that these types of intrinsically motivated behaviours are crucial for an agent to gain the competence necessary to achieve the level of control and mastery over its environment that it requires for autonomy. Similar approaches instead focussed on a “knowledge-based” view of intrinsic motivation[17][18]. Whilst clearly these are closely related, we prefer to focus on the competence-based view as an agent which aims to maximise its competence will naturally be required to seek out more knowledge about its environment. It is not quite so clear that an agent aiming to maximise knowledge must necessarily become competent in its environment as well, although this may also be true.

The link to the principle of future state maximisation is that any agent which aims

to maximise the number of possible actions that it can carry out in the future necessarily needs to be competent enough to perform a large variety of manipulations on its environment. To do this well would require both knowledge of how the environment works, as well as a broad set of reusable skills that can be applied to solve the variety of different problems that it might encounter during its quest to maximise its future options. Thus we can view FSM as an intrinsic motivation which provides an incentive for an agent to act in situations when there is no immediate external reward, or no immediate specific goals to be achieved, by driving it to increase the amount of control that it has over its future. In general, the more control the agent has the better placed it will be to achieve any number of potential goals it could find itself with later on. Indeed, we can also imagine that the very process of aiming to maximise its future possibilities could naturally lead to an agent spontaneously setting goals for itself. Taking a very high-level viewpoint, if an agent reasons, say, that it can increase its number of future possibilities by going to university, then it will set itself the goal of finishing university. This will then include many shorter term goals, e.g. passing particular classes and exams.

2.2 The “empowerment” framework

Ideas which can be loosely related to the FSM principle have appeared previously in various different contexts. For example, many chess (and other game) playing algorithms use the concept of mobility, which is the number of moves available to a player, as part of an evaluation function to score the desirability of any given game state. This represents the intuitive idea that situations in which a player has a lot of different moves available to them are likely to be preferable to situations with less choice. However, in this context it was never associated with a more rigorous theoretical foundation. Introduced by Klyubin et al. in 2005[5][19], the concept of “empowerment” represents the first attempt that we are aware of to create a formal, quantitative framework directly related to this principle of FSM. It does so using the language of information theory, essentially seeking to quantify the total amount of *potential* influence that an agent’s actions have over its environment at some time in the future. The quantity defined is universal in the sense that it can be applied to any agent-environment interaction which can be expressed probabilistically, and task-independent in that it can be applied to any situation without the need to specify a particular goal.

2.2.1 Formalism

Before describing how the empowerment measure is defined it is necessary first to introduce some fundamental concepts from information theory. The Shannon entropy[20] H associated with a discrete random variable X , which can take values $x \in \mathcal{X}$, is defined to be:

$$H(X) = - \sum_{x \in \mathcal{X}} \mathbb{P}(x) \log[\mathbb{P}(x)] \quad (2.1)$$

where $\mathbb{P}(x)$ is the probability of observing the value x and generally the base of the logarithm is taken to be 2, in which case we say that this is measured in bits. This quantifies the amount of uncertainty associated with making a measurement of X , or equivalently the average amount of information that we obtain when we make a measurement of X . If we consider another random variable Y which can take values $y \in \mathcal{Y}$ we can define the conditional entropy as follows:

$$H(X|Y) = - \sum_{y \in \mathcal{Y}} \mathbb{P}(y) \sum_{x \in \mathcal{X}} \mathbb{P}(x|y) \log[\mathbb{P}(x|y)] \quad (2.2)$$

This provides a measure of the average amount of extra information gained when making a measurement of X , given that you have the result of measuring Y , and allows us to define a symmetric quantity known as the *mutual information*:

$$I(X; Y) = H(Y) - H(Y|X) = H(X) - H(X|Y) \quad (2.3)$$

This quantifies the average amount of information that is gained about X , *solely by measuring* Y , and vice-versa.

The classical communication problem in information theory involves imagining a “sender” who transmits a message to a “receiver” via an abstract “channel”, which in general is noisy. If we take X to be the random variable associated with the sender’s input to the channel, and Y to be the random variable associated with the channel’s output, then the *channel capacity* C is defined as the maximum possible mutual information between the input and the output signals:

$$C(X \rightarrow Y) := \max_{\mathbb{P}(x)} I(X; Y) \quad (2.4)$$

where here the maximisation is over all possible distributions of the input signal. This represents the maximum average amount of information that the received signal can

possibly contain about the input signal. In the context of empowerment we consider an agent equipped with some kind of sensors. Suppose that the set of actions it is able to choose from at any given time is \mathcal{A} , and the set of possible states its sensors can be in is \mathcal{S} . For simplicity, we consider only discrete time steps and define A_t to be a random variable associated with the action the agent chooses at time step t , and similarly S_t for the agent's sensor states (with a_t and s_t representing the actually realised values). We then define a vector of random variables associated with a sequence of n actions, $A_t^n = (A_t, A_{t+1}, \dots, A_{t+n-1})$. The n -step empowerment \mathfrak{E}_n can now be defined in terms of the channel capacity between the set of available action sequences of length n and the set of possible sensor states at a later time $t + n$:

$$\mathfrak{E}_n := C(A_t^n \rightarrow S_{t+n}) \equiv \max_{\mathbb{P}(a_t^n)} I(A_t^n; S_{t+n}) \quad (2.5)$$

That is, we consider the empowerment to be the maximum amount of *potential* information that an agent can transmit to its sensors at a later time, given the set of possible action sequences it can potentially take. This is a direct way of measuring the amount of control the agent has over what states its future sensors can possibly read; the more reliably an agent's actions are able to influence the value of the future sensor states, and the more variety of different sensor states which can be reached, the higher the agent's empowerment will be — i.e. if many different action sequences tend to lead to the same sensor states, this will lead to the empowerment value being lower than if they were to lead to many different possible states. It is worth emphasising at this point that the calculation of this quantity involves considering all actions that the agent *can possibly* do, rather than only the actual trajectory that it will eventually follow. This is because we are looking for the probability distribution over all possible action sequences which allows for the largest amount of potential information transfer to the agent's sensors at a later point in time. As such, this generally makes the calculation of empowerment very computationally expensive, especially as n is increased (established methods for calculating channel capacities between discrete random variables, such as the Blahut-Arimoto algorithm [21, 22], involve estimating an initial source distribution and iteratively improving it over a number of steps. Here this would require considering every possible action sequence of n -steps *multiple* times, making it extremely slow if the state/action spaces are large). It is also worth noting how this framework refers explicitly to an agent's sensors, such that it is only measuring the amount of control the agent has over its environment which it can itself perceive. This means it does not really matter how much influence the agent has from an outside point of view, and that the empowerment

quantity is localised in the sense that it is based solely on the perception-action loop of the agent.

One special case where things simplify significantly and which is particularly relevant for the work described in Chapter 4 is when we have a deterministic environment and a discrete, finite set of actions and sensor states. In this case, each of the i possible action sequences available at time t , $a_{i,t}^n$, will lead to exactly one sensor state $s_{i,t+n}$, i.e.

$$\mathbb{P}(s|a_{i,t}^n) = \begin{cases} 1 & \text{if } s = s_{i,t+n} \\ 0 & \text{otherwise} \end{cases} \quad (2.6)$$

Consequently it is clear that $H(S_{t+n}|A_t^n) = 0$, since the action sequence fully determines the later sensor state. As such, no additional information is gained when you observe the sensor state given that you already know what the action sequence was. This means that in this case the empowerment reduces to:

$$\mathfrak{E}_n = \max_{\mathbb{P}(a_t^n)} H(S_{t+n}) \quad (2.7)$$

i.e. the maximum Shannon entropy over the agent's future sensor states. This is achieved for the distribution over action sequences which leads to a uniform distribution over the possible sensor states \mathcal{S}_{t+n} [23] which are accessible at time $t+n$. Consequently:

$$\mathfrak{E}_n = - \sum_{s \in \mathcal{S}_{t+n}} \frac{1}{|\mathcal{S}_{t+n}|} \log \left(\frac{1}{|\mathcal{S}_{t+n}|} \right) = \log (|\mathcal{S}_{t+n}|) \quad (2.8)$$

where $|\mathcal{S}_{t+n}|$ is the size of the set \mathcal{S}_{t+n} , i.e. the empowerment reduces to the logarithm of the number of distinct sensor states that can be reached with the available set of action sequences.

2.2.2 Previous work

Due to the computational cost associated with calculating an agent's empowerment, applications of this framework have generally been confined to studying simple toy-systems such as mazes and small grid-worlds[19]. This has involved studying both the basic behaviour which emerges from empowerment maximisation in these scenarios as well as using empowerment directly as a fitness function, running evolutionary algorithms to evolve the types of sensors and actions that the agents possess[24]. Other work has

looked at ways of approximating the empowerment in a more computationally efficient manner. The difficulty is that one has to consider every action that an agent can possibly take, which grows exponentially with the length of the time horizon that we consider, i.e. the number of time steps into the future that we choose to model. One early approach which seeks to alleviate this somewhat for systems with discrete action/state spaces is known as “impoverished empowerment”. Here the idea is to evaluate only a limited subset of action sequences — selecting those which contribute most significantly to the total empowerment[25]. This allows for longer action sequences to be built iteratively whilst still providing a reasonable approximation to the full empowerment in many situations. A more recent approach uses a modification of the Monte Carlo Tree search technique to speed up the computation of empowerment in the case where we have a deterministic environment[26]. This allows for exploration further into the future by biasing the search towards future action sequences which have already shown promise in terms of contributing significantly to the agent’s empowerment. Another approach derives a lower-bound for the mutual information, which can be calculated much more efficiently, and then maximises this instead [27]. This work also makes a link between empowerment and intrinsic motivation, using the derived empowerment approximation as an intrinsic reward for some simple reinforcement learning tasks.

The case where we have continuous sensor states or action spaces has also been looked at, however it is significantly more difficult since in general there is no known solution to computing the channel capacity of a continuous channel. Furthermore, a naive approach of discretising the continuous space into bins does not work well since the state space quickly becomes too large to be tractable for anything but the most trivial examples. Another problem arises with a direct generalisation of the framework to continuous spaces when the environment is deterministic (i.e. no noise), since it is then possible for the channel capacity to become infinite. This is because one can store an arbitrary amount of information in an infinite precision (noiseless) real number[23]. Of course this can always be overcome by adding in some small amount of noise, and indeed any realistic situation will include some amount of noise anyway. One approximate method which has been applied to some simple continuous systems is the “Monte Carlo Integration Method”[28], although this requires making a number of approximations about the distribution of sensor states for a given action sequence. Consequently the continuous case remains significantly more difficult to deal with than with the discrete sensor/action case.

Finally, we mention some work on multi-agent empowerment. In situations where agents

share an environment their empowerments become naturally intertwined. This was investigated by Capdepuy et al.[29] who looked at agents occupying a simple grid world, equipped with basic sensors to detect the density of other agents around them in different directions. Here each agent was encoded to selfishly maximise its own empowerment, requiring a balance between staying in the vicinity of other agents whilst also remaining sufficiently distant so as to maintain the freedom to move around without becoming trapped. They found that this tension naturally led to the formation of non-trivial, dynamical structures. In other simple models studied[30][31] they found examples analogous to a number of situations which arise in game theory. For example, in some situations the empowerment of one agent could only be increased at the expense of decreasing the empowerment of the others (like a zero-sum game), whereas in other situations the selfish maximisation of empowerment could lead to increased empowerment for the other agents too.

2.3 The “causal entropic forces” framework

Another separate attempt to formalise this idea of FSM is the “causal entropic forces” framework, introduced by Wissner-Gross and Freer[6] in 2013. This takes a much more thermodynamic approach by defining an entropy over all of the possible paths that a system can follow for some time into the future, τ , and then generating a force which drives the system towards states which have a larger value of this entropy. Rather than considering the microstate of a system as being defined by a point in phase space, instead they consider a microstate to be a trajectory through phase space $\mathbf{x}(t)$ over some future time interval, $0 \leq t \leq \tau$. These can then be partitioned into “macrostates” based on the initial position in phase-space that the paths start at, \mathbf{X} . That is, we consider two microstates $\mathbf{x}(t)$ and $\mathbf{x}'(t)$ to belong to the same macrostate \mathbf{X} if (and only if) $\mathbf{x}(0) = \mathbf{x}'(0)$. This allows us to define the “causal path entropy” associated with the current macrostate \mathbf{X} (i.e. the current position in phase-space) as a path integral over future paths:

$$S_c(\mathbf{X}, \tau) = -k_b \int_{\mathbf{x}(t)} \mathbb{P}[\mathbf{x}(t)|\mathbf{x}(0)] \log (\mathbb{P}[\mathbf{x}(t)|\mathbf{x}(0)]) \mathcal{D}\mathbf{x}(t) \quad (2.9)$$

where here $\mathbb{P}[\mathbf{x}(t)|\mathbf{x}(0)]$ is the probability of observing the path $\mathbf{x}(t)$ under the system’s normal dynamics (i.e. without the inclusion of an entropic force). In effect, this is providing a measure of the number of potential possible paths the system has available to it given its current position, out to a time τ into the future. It is then proposed that

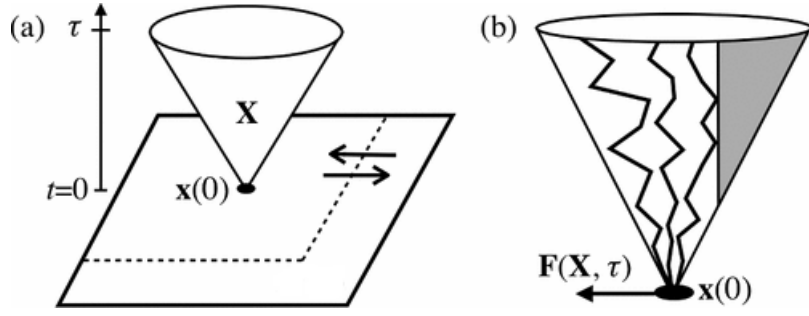


Figure 2.1: Taken with permission from Wissner-Gross and Freer (2013)[6]. Visualization of a causal entropic force. (a) The cone represents all of the future paths which belong to macrostate \mathbf{X} starting from $\mathbf{x}(0)$. (b) Here we imagine an environmentally imposed constraint which excludes some region of future-path phase-space, e.g. with the presence of a wall, which excludes the region shaded in grey. This breaking of symmetry will induce a force which drives the system to regions of greater freedom.

an “intelligent” system (or at least a system which wishes to maximise its number of future paths) should follow a gradient in this entropy, and hence proposes the following definition of a “causal entropic force”:

$$\mathbf{F}(\mathbf{X}_0, \tau) = T_c \nabla_{\mathbf{X}} S_c(\mathbf{X}, \tau)|_{\mathbf{x}_0} \quad (2.10)$$

Here T_c is a parameter which is somehow analogous to a temperature, determining how strongly the system is driven towards states of higher future path entropy, whilst \mathbf{X}_0 is the current state of the system. This idea is illustrated schematically in figure 2.1.

As with empowerment there are difficulties in calculating this quantity since it involves taking the derivative of a path integral over all of the possible paths that a system can follow for some time into the future — and overall much less work has been done in looking at ways to speed up or approximate the calculation of a causal entropic force. As such, the examples studied have again been simple toy systems, such as a cart pole system which under causal entropic forcing naturally swings up to stabilise itself, since this provides it with the easiest access to the largest variety of possible states. Another example studied was a simple puzzle where a large circular agent has to make use of a smaller circle (a tool) in order to free another smaller circle which is otherwise inaccessible. In this case, the larger circle is incentivised to do this because, once the smaller circle has been freed there are now two circles it is able to manipulate — meaning

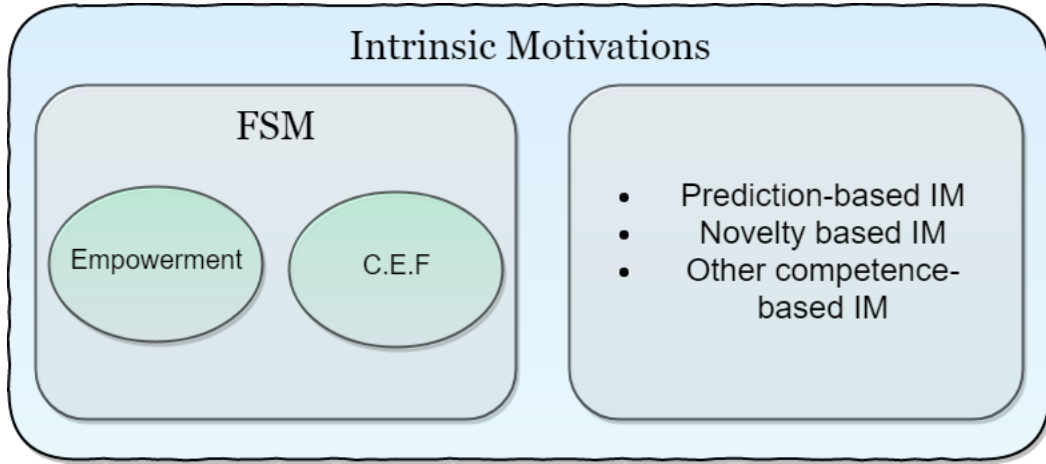


Figure 2.2: A high-level summary of the relationships between the concepts described in the chapter so far. We view FSM as a subset of intrinsic motivations as it can be used as the basis for generating behaviour even when an agent has no immediate goals to achieve or any expectations of receiving a well-defined reward. We can think of empowerment and causal entropic forces (C.E.F) as particular frameworks that allow us to implement the underlying philosophy of FSM in a quantitative way.

that it has combinatorically more paths available to it.

Figure 2.2 gives a high-level summary of the relationship between intrinsic motivation, FSM, empowerment and causal entropic forces. We argue that FSM can be thought of as the underlying philosophical framework for both empowerment and causal entropic forces. That is, the philosophy that in the absence of a particular goal an agent should make decisions to increase its control over the potential future states it can access is what we should think of as FSM — with empowerment and causal entropic forces being two particular ways in which this principle can be implemented in a quantitative manner. We can then think of FSM as a whole to be a subset of intrinsic motivations, since it can be used as the basis of making decisions in the absence of any expectations of a well-defined “reward”.

2.4 Collective motion

In this thesis, our primary application of the FSM principle is to a model of agents equipped with simple visual sensors that leads to collective motion(see Chapter 4). Here we give a brief overview of some of the previous work done in this field and discuss the

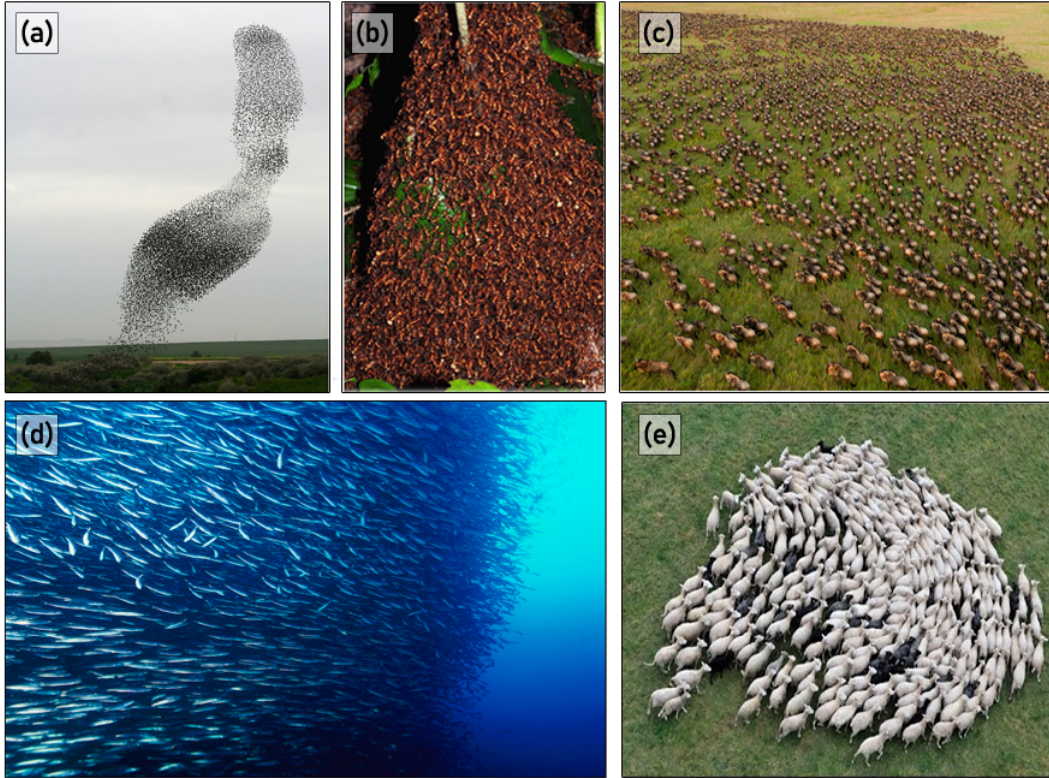


Figure 2.3: Examples of collective motion throughout nature. (a) A Murmuration of starlings. (b) A colony of army ants. (c) A herd of wildebeest. (d) A school of fish. (e) A herd of sheep. All images were sourced from Wikimedia Commons.

most relevant results. We refer the reader to a review by Vicsek and Zafeiris[32] for a more detailed account.

Collective motion is a phenomenon which occurs throughout the natural world at many different scales, with examples ranging from huge murmurations of starlings made up of hundreds of thousands of individual birds down to the level of colonies of bacteria (see figure 2.3 for images of a number of different systems). The characteristic feature of collective behaviour is that the actions taken by any given individual are almost completely dominated by the influence of the other agents around it, i.e. the agents behave in a completely different manner when they are together in a group than they do when they are alone. This often leads to complex and visually impressive behaviour. A number of studies have been carried out where detailed observations and measurements of a variety of swarming systems have been made, including bacteria[33], insects[34][35], fish[36] and

birds[37][38]. The evolutionary advantages of this collective “swarming” behaviour are thought to include benefits such as improving the collective awareness of the group, due to the sharing of information, and providing additional protection from predators[39]. Trying to understand how these systems are able to exhibit the collective behaviour that we observe is a fundamentally interesting scientific question in itself. It is also possible that understanding how these natural systems work could also be useful in building artificial systems made up of many interacting components. One hope is that these may be useful in solving difficult problems efficiently in a collective, decentralised manner (this idea is the basis for the field of “swarm robotics” [40]).

One of the best ways to gain a better understanding of how a system works is to try and build a minimal model which can reproduce the key features of the observed behaviour. One of the earliest attempts made to create a model of collective motion was the “Boids model” developed by Reynolds in 1987[41]. This model involved a local alignment interaction between agents along with long-range attraction and short-range repulsion to keep the flock cohesive. This was used in the context of computer animation as an alternative to having to code the trajectory of each agent by hand. A similar model was introduced later on by Vicsek et al. in 1995 taking a more quantitative, statistical physics based approach. Here each agent updates its orientation at each time step by averaging over all of its neighbours within a fixed distance R , as well as adding some noise. The agents are all constrained to move at a constant speed v_0 and so the equations of motion for agent i ’s orientation θ_i and position \mathbf{r}_i are given by:

$$\begin{aligned}\theta_i(t+1) &= \langle \theta_k(t) \rangle_{|\mathbf{r}_k(t) - \mathbf{r}_i(t)| < R} + \eta_\theta(t) \\ \mathbf{v}_i(t+1) &= \begin{bmatrix} \cos \theta_i(t+1) \\ \sin \theta_i(t+1) \end{bmatrix} \\ \mathbf{r}_i(t+1) &= \mathbf{r}_i(t) + \mathbf{v}_i(t+1)\Delta t\end{aligned}\tag{2.11}$$

where $\langle \theta_k(t) \rangle_{|\mathbf{r}_k(t) - \mathbf{r}_i(t)| < R} = \arctan \frac{\langle \sin \theta_k(t) \rangle_{|\mathbf{r}_k(t) - \mathbf{r}_i(t)| < R}}{\langle \cos \theta_k(t) \rangle_{|\mathbf{r}_k(t) - \mathbf{r}_i(t)| < R}}$ and $\eta_\theta(t)$ is an uncorrelated random number on the interval $[-\eta/2, \eta/2]$ at each time step and Δt is the duration of a single time step.

The model attracted a lot of attention as it was the first attempt to establish a quantitative model for the behaviour of large flocks of self-propelled individuals which interact via simple rules. In addition they were also able to demonstrate that there was a second-order phase transition between a globally disordered and a globally ordered phase, as a

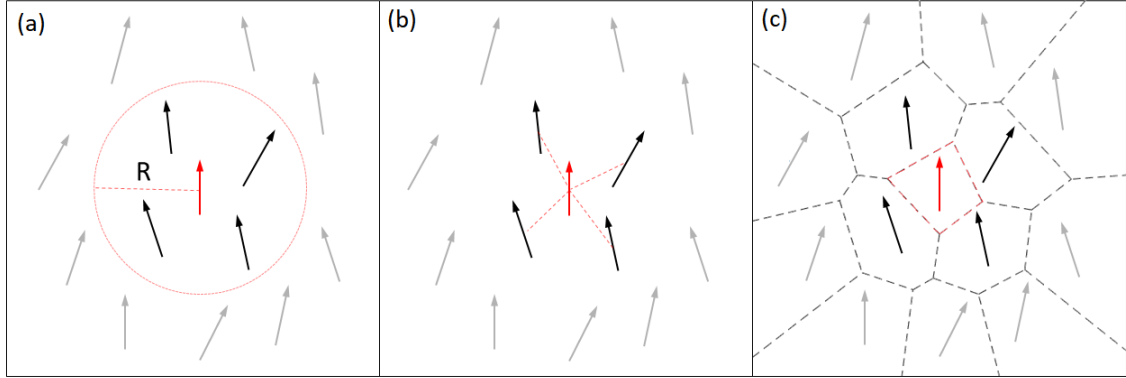


Figure 2.4: Neighbour selection strategies for variations of the Vicsek model. Agents update their orientations by averaging over the orientations of neighbours. (a) Standard Vicsek Model: neighbours are those within a fixed distance R . (b) Metric-Free Vicsek Model: a certain number of nearest neighbours regardless of how far away they are. (c) Topological Vicsek Model: a Voronoi tessellation identifies neighbours in a topological manner.

function of either the flock density or the noise parameter η . This was studied in more detail by Chat   et al.[42] where the nature of the transition was disputed and shown to be discontinuous under a range of conditions.

Whilst the Vicsek model is structurally simple and not intended to be a realistic model for the collective motion of animals per se, it nevertheless demonstrated that a simple local co-alignment rule was sufficient to generate a globally ordered flock. Certain modifications to improve the Vicsek model were inspired by data taken by the STARFLAG collaboration, where detailed measurements of many real starling murmurations of up to a few thousand individuals were made[38][37]. Their statistical analyses determined that starlings seem to interact with a fixed number of nearest neighbours, regardless of the actual distances between the birds. They also observed that the velocity correlation length between individuals scaled with the size of the flock. This makes them “scale-free”, in that there is no characteristic length scale over which individuals remain correlated. This is thought to allow for a significantly faster collective response to external perturbations[37] (e.g detecting a predator). The simplest way to modify the Vicsek model to make it “metric-free” is to simply average the orientation over the k -nearest neighbours[38], rather than the neighbours within a fixed radius R (see figure 2.4(b)). A different approach which is completely metric-free is the “topological Vicsek model”[43], which selects neighbours by creating a Voronoi tessellation[44] and assigning neighbours

based on whether their cells are touching (see figure 2.4(c)). This has the feature that it is no longer necessary to specify a parameter for the number of nearest-neighbours to interact with. In both of these models the resulting velocity correlations are also found to be scale-free.

A problem with many variations of the Vicsek model, already mentioned, is that they do not naturally remain cohesive, i.e. flock’s density will gradually decrease indefinitely towards zero. This is often handled by putting the system in an artificial “box” with periodic boundaries, however as we look towards creating more realistic models this is obviously not satisfactory since the sky is essentially unbounded. One solution is to use a pairwise motional bias between agents, attractive at large distances and repulsive at short distances, as was used in the original Boids model. However this introduces a metric for the preferred separation of agents, meaning that the velocity correlations between agents will no longer be scale-free. One model which addresses this is the “Strictly Metric Free Model” [45] which uses the Voronoi tessellation to determine which agents are on the “edge” of the flock in a metric-free manner. These agents then have an inward motional bias added to them, keeping the flock cohesive. Another model, known as the “Hybrid Projection Model” [46], instead keeps the flock cohesive by including a term based on the visual projection of the other agents alongside the local co-alignment with topological neighbours. This is one of the first models which explicitly includes long-range visual interactions, which are likely to be important in understanding the collective motion of a number of higher biological organisms. Despite the successes of these models in explaining various aspects of the observed data they are all essentially empirical — including either explicit co-alignment or cohesion rules hard-wired into them. As such they do not provide significant low-level explanatory power for how and why agents actually co-align and remain in cohesive groups — a point which we aim to address with our model in Chapter 4.

Chapter 3

Quantifying future state maximisation

In this chapter we think about different ways in which the principle of FSM can be quantified. We start by considering systems which have a discrete and finite set of states, where the transitions between states can be modelled on a graph. We show that for these simple systems there is a straightforward and efficient way to calculate both the number of paths available as well as the number of end states which can be reached from the current state, looking n steps into the future. These quantities can be related to causal entropic forces and empowerment respectively, and we will discuss this connection in detail. Whilst generally it is not possible to increase n without bound within these frameworks, we show that for these simple systems large n can be handled without difficulty. This provides us with a set of examples in which we are able to probe the large future time horizon limit.

After this, we introduce a new quantity based on comparing the average difference between possible future states which are available. Whilst this does not have as strong a theoretical foundation as empowerment it nevertheless shares the basic philosophy of FSM, and is often more straightforward to apply to continuous state and action spaces.

3.1 Discrete path/final state counting with adjacency matrices

Consider a system which can be described by a graph (directed or undirected), where each state that the system can occupy is represented as a node and allowed transitions

between states are represented as edges. If there are N states that the system can be in we can define an $N \times N$ matrix A , called the adjacency matrix, in the following way:

$$A_{i,j} = \begin{cases} 1 & \text{if there is an edge between node } i \text{ and node } j \\ 0 & \text{otherwise} \end{cases} \quad (3.1)$$

This contains all of the structural information about the graph. We can then use this matrix to count the number of paths of length n between node i and node j , allowing for paths which go back on themselves and visit the same node multiple times:

$$N_{paths}^{(n)}(i, j) = \sum_{d_1} \sum_{d_2} \cdots \sum_{d_{n-1}} A_{i,d_1} A_{d_1,d_2} \cdots A_{d_{n-1},j} = (A^n)_{ij} \quad (3.2)$$

such that the total number of paths of length n starting from state i is given by:

$$N_{paths}^{(n)}(i) = \sum_j (A^n)_{i,j} \quad (3.3)$$

The philosophy of moving so as to maximise the number of options available in the future can then be implemented by transitioning to the neighbouring state which has the largest number of future paths available to it, i.e. essentially following a gradient in the number of future paths, analogous to the causal entropic forces framework. That is, the dynamics on the graph are generated at each time step by transitioning to the node with the highest number of paths available to it (which is connected to the current node).

Alternatively, we can just consider the number of possible future states that are accessible after n steps rather than the total number of paths. This is more similar to empowerment, as although empowerment is introduced by talking about an agent's "sensors" there is nothing stopping us from defining a sensor to be a complete specification of the state of the system (or to use a graph that models the transitions between sensor states). We can obtain this in the same way for each state by taking A^n , then counting the number of entries that are more than zero. That is, if we define:

$$A_{i,j}^{*(n)} = \begin{cases} 1 & \text{if } (A^n)_{i,j} > 0 \\ 0 & \text{otherwise} \end{cases} \quad (3.4)$$

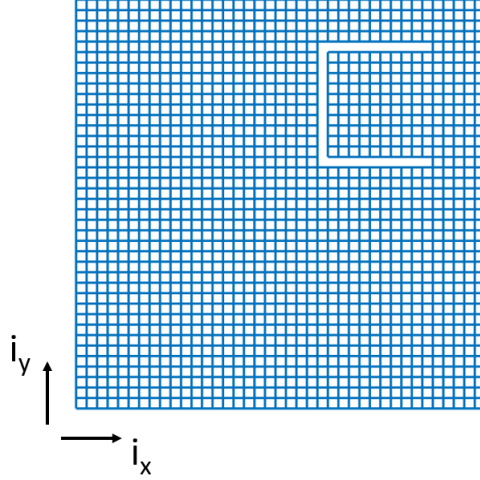


Figure 3.1: As a simple toy system to study we consider a 40×40 square non-periodic lattice that has had some edges removed from it. Here a state $\mathbf{i} = \{i_x, i_y\}$ is defined in terms of its x and y coordinates, i_x and i_y respectively. Nodes on the graph are considered to be neighbours if there is an edge connecting them on the lattice. Heat maps showing the number of future paths/accessible end states from each initial state are shown in figures 3.2 and 3.3.

then the number of possible states that can be accessed from state i after n steps is just:

$$N_{states}^{(n)}(i) = \sum_j A_{i,j}^{*(n)} \quad (3.5)$$

In the case of a deterministic system where the transitions are between sensor states this quantity can be directly related to the n -step empowerment:

$$\mathfrak{E}_n(i) = \log \left(N_{states}^{(n)}(i) \right) \quad (3.6)$$

As a simple example of putting this into action and evaluating how the two approaches differ, we consider a graph made up of a square 40×40 lattice where some edges have been removed, as shown in figure 3.1. To fully represent the allowed transitions between states requires a 1600×1600 adjacency matrix A . Whilst initially A is very sparse it should be noted that this does not remain true of A when taken to large powers, demonstrating how the memory requirements of these methods scale badly as the size of the system is increased. The set up is such that the region of largest “freedom”, that is where the most number of paths are available, is near the middle (away from the walls) but biased slightly towards the bottom left due to the edges which have been removed.

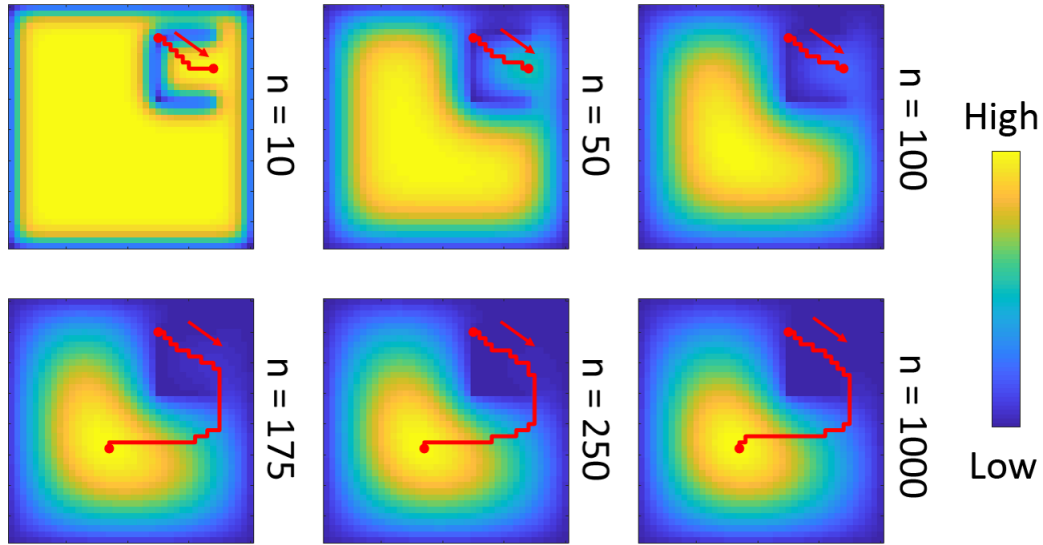


Figure 3.2: Heat maps showing the number of future paths of varying lengths, available from each point on the lattice shown in figure 3.1. The red line shows the path taken by following the gradient in the number of future paths from the initial point inside the “potential well”, and the arrows indicate the direction. We see that when n is made large enough the system is able to escape over what is effectively a potential barrier, reaching the region with the largest freedom. Once a value of n is found in which the well is escaped from, this remains the case for all larger values of n too.

The heat maps in figure 3.2 provide a representation of the number of paths of a given length n that are available from each starting point. The red lines show the path taken if we follow a gradient in this measure, towards states which act as a starting point for a larger number of paths, starting from a state with particularly low freedom inside the C shape.

We see that to escape from the kind of potential well that has been created in the top right requires long paths. In fact, for this starting point we find empirically that for all $n < 154$ the red line always gets “stuck” inside the well, whereas for $n \geq 154$ it is always able to escape. This suggests that there may be situations when using these methods with the deliberate addition of stochasticity could be useful to aid in escaping potential barriers (e.g. for $n = 50$ and $n = 100$, we clearly see that there is a region of much greater freedom which cannot be reached by following a local gradient). We can compare this to what happens if we just count the number of endpoints after n moves, shown in figure 3.3.

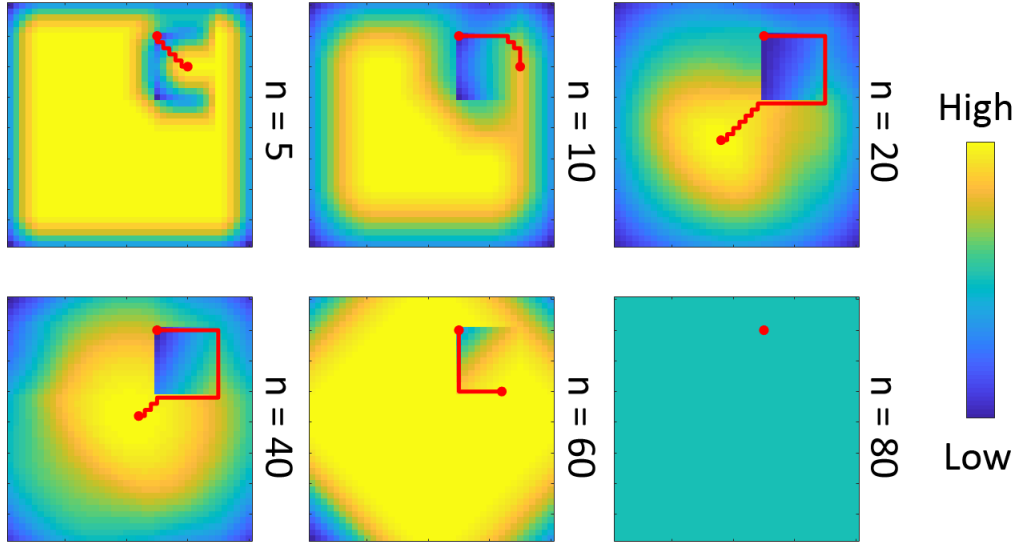


Figure 3.3: Heat maps showing the number of states accessible after n steps, starting from any given initial state on the lattice shown in figure 3.1. As with figure 3.2, the red line shows the path followed by following a gradient in this measure. We see that, for a range of n values (which are much lower than required in the path counting example), the system successfully escapes into the region where intuitively the “freedom” is higher. However, if we continue to make n even larger eventually this stops and every initial state begins to look identical under this measure. This happens when *every state is accessible from every other state*.

This example is similar to that considered by Klyubin et al.[5] in their initial paper on empowerment. We find that we sometimes get the same qualitative behaviour as with the path counting approach, in that a gradient can be formed to “escape” from the potential well (although it is interesting to note that this occurs by following the walls of the potential well rather than being repelled from them as in figure 3.2). This occurs initially for n somewhere between 10 and 20, a significantly lower value than when we count paths.

However, as n is increased further this eventually stops, and by the time that $n = 80$ we have a completely flat empowerment landscape. This is because, at this point, every state is accessible from every other state within 80 moves, such that there is no longer anything to distinguish between the desirability of locations under this measure. This has been discussed in the empowerment literature[23], and means that one has to be careful to choose a sensible value for n . Although in many situations it may be fairly simple to intuitively use a “reasonable” value, it’s not obvious what a formal criterion should be for determining this in advance. This could be counted as a potential advantage of the path counting approach (at least on a discrete graph), since it does not suffer from this issue.

This advantage, however, is primarily theoretical since n is the quantity which is most restricted (as the number of future possibilities increases exponentially). As such, we might not usually be concerned about it becoming too large. It is only for the special case we are considering, where we can represent the system on a graph and calculate the number of paths/end points by taking powers of the adjacency matrix, where increasing n does not lead to a significant increase in computation. We can analyse this in a bit more detail — if A is an $N \times N$ matrix, then taking A^n takes $O(M(N) \log(n))$ operations, where $M(N)$ is the number of operations required to perform the matrix multiplication of two $N \times N$ matrices (we have $\log n$ rather than n since taking the matrix power can be broken down into stages. For example, if we take A^{16} , we can say $A^{16} = A^8 \times A^8$, $A^8 = A^4 \times A^4$, $A^4 = A^2 \times A^2$ and $A^2 = A \times A$, meaning we only need to do $4 = \log_2 16$ matrix multiplications). Consequently, considering paths of much longer lengths does not significantly increase the computational time, although as a matter of practicality we still have to be careful with the numerical precision being used, as the numbers involved quickly become very large as n is increased. Perhaps the biggest issue with these methods is the memory needed to store the adjacency matrix (or more significantly,

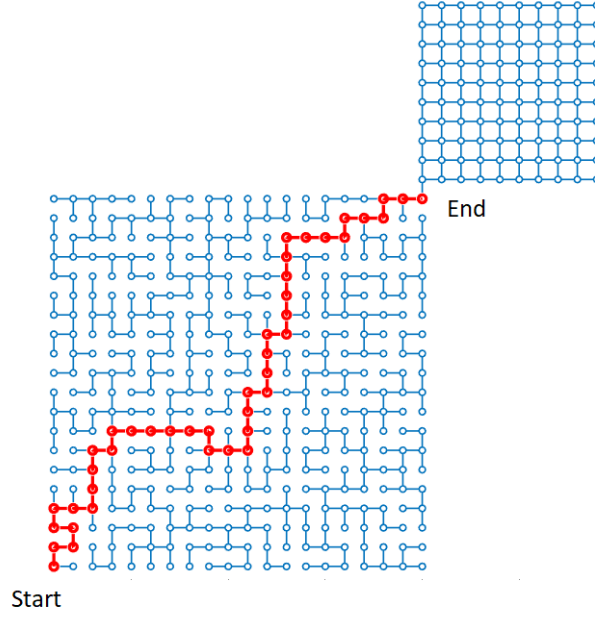


Figure 3.4: Example of a 20×20 maze connected to a 10×10 periodic lattice at its end point. This is a region of “higher freedom” than anywhere within the maze. The path in red shows the only distinct solution (which does not go back on itself), starting from the point in the bottom left and ending at the top right.

powers of the adjacency matrix which are much less sparse). This requires storing N^2 integers, where N is the number of states. For virtually any complex system of interest, N grows extremely quickly. For example, consider an $N_L \times N_L$ lattice occupied by n_A agents which are free to move around. If each agent is allowed to occupy any of the lattice sites then the total number of states the system can be in is given by $N = N_L^{2n_A}$. This quickly becomes unmanageable for large N_L or n_A .

3.1.1 Maze solving application

As a second application we now consider applying these methods to solving a maze, i.e. finding a path from the entrance of a maze to the exit. If we consider a square grid of size $N_m \times N_m$, a “perfect maze” is defined as a spanning tree of this grid. This means that between every two points there exists exactly one direct path that connects them.

We make use of an elegant algorithm by Wilson that generates perfect mazes uniformly from the set of all possible spanning trees of a square grid, using loop-erased random

walks[47]. We then arbitrarily choose the bottom left corner to be the beginning of the maze and the top right corner to be the endpoint. We connect this endpoint to a second region of state space which is more highly connected than anywhere within the maze itself, e.g. a fully locally connected periodic lattice, as shown in figure 3.4. We refer to this as a “freedom reservoir”.

Providing that the freedom reservoir is more highly connected than the maze and sufficiently large then the path counting approach described in the previous section will eventually lead to a gradient in number of future paths which “solves” the maze, providing a path which drives the system from the start point into the freedom reservoir. Indeed, when n is made large enough this is exactly what we find. How far into the future we need to look before successfully navigating out of the maze of course depends on the size of the maze, but also on the size and connectivity of the freedom reservoir to which it is connected. This is investigated in figure 3.5, where we look at the distribution (as the mazes are generated randomly) of minimum future path lengths required to generate a gradient path which solves the maze, for a variety of choices of freedom reservoir. For comparison, we also plot the distribution of the actual solution lengths, i.e. the minimum number of steps required to get from the start of the maze to the endpoint. The inset figure shows a scatter plot of the minimum future path length (n) vs. the maze solution length for two types of freedom reservoir; a lattice and a set of fully connected nodes.

Probably the most natural choice for a freedom reservoir is a highly connected lattice. We include results for a 5×5 as well as a 20×20 lattice in figure 3.5. Because these are periodic, since this allows for a larger number of paths to be generated within the reservoir, it should not be surprising that changing the lattice from 5×5 to 10×10 (or anything larger) does not significantly alter the results. We can however try connecting the end point to a fully connected graph, i.e. a graph where every node is connected to every other node. Clearly this means there will be more paths available within the freedom reservoir, and we see that this leads to the distribution of the minimum future path length being shifted to smaller values of n . In fact, as the number of fully connected nodes goes to infinity, we should expect that when n equals the minimum maze solution length then there should be a strong enough freedom gradient to drive the system out of the maze. Looking at the inset of figure 3.5, we see that this is almost the case when we attach the end point to 100 fully connected nodes.

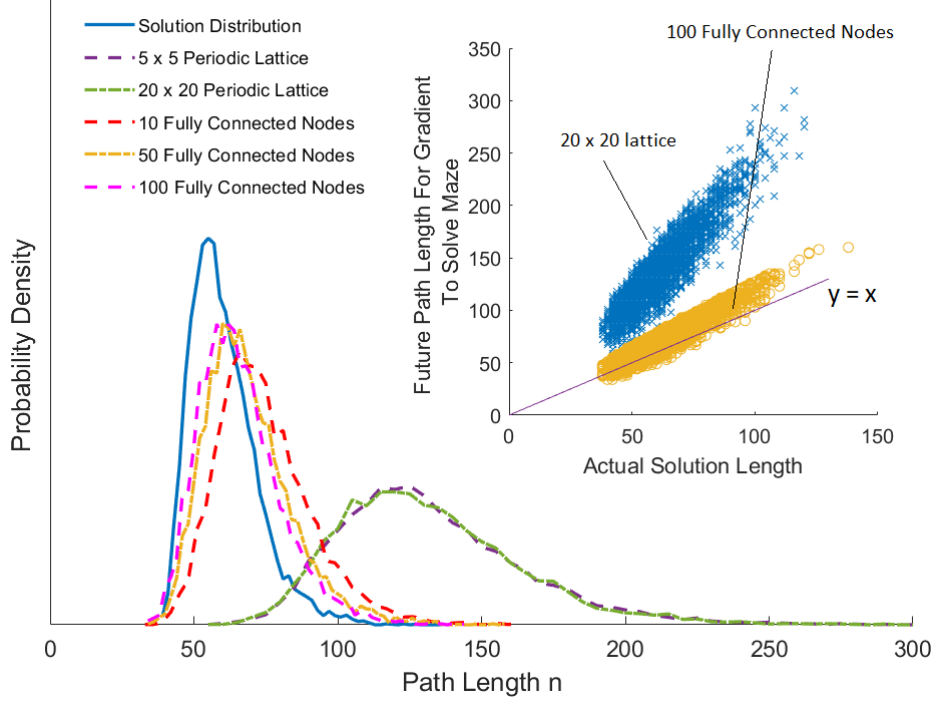


Figure 3.5: The distribution of future path lengths n required to solve a maze, for different “freedom reservoirs”. The solid blue line shows the distribution of the actual length of the solution (the minimum number of steps needed to get from the start of the maze to the end) for randomly generated 20×20 mazes, whilst the other curves show the distribution of the *minimum future path length* that must be considered before following the gradient in number of future paths solves the maze. Each of these uses data gathered from 10000 different mazes. The inset shows a scatter plot of minimum future path length vs. actual solution length. We see that when we connect the end point of the maze to a very highly connected reservoir the line approaches $y = x$, i.e. the maze will be solved as soon as any of the future paths from the starting point of the maze can “feel” the freedom reservoir.

Next we consider the result of a similar analysis using the maximisation of potential future end states after n steps instead (equation 3.5). In this case we have to be more careful about the freedom reservoir we choose to connect the end point to. Firstly, the size of the lattice is much more important, and it is no longer sensible to use a periodic lattice. This is because once all of the sites in the freedom reservoir are accessible, increasing n further will not make any difference, and using a periodic lattice allows exploration of the full reservoir in a smaller number of steps. In fact, to reliably solve a 20×20 maze requires connecting the endpoint to a lattice which is roughly 50×50 , otherwise the system tends to get stuck in local minima in number of potential end states (regardless of how large n is made). This means that we need a much larger adjacency matrix to fully describe the system, and we still have the problem that even when n is large enough to solve the maze if it is increased further eventually all initial states are equally preferable under this measure. Nevertheless, we again find that the “intelligent behaviour”, i.e. solving the maze, can be made to occur for smaller values of n . Indeed, if it happens at all, then it will happen for significantly lower n . For example with $N_m = 20$ and connected to a 50×50 lattice, on average we have to use future path lengths of ≈ 48 before the freedom gradient solves the maze. Using the same set up with the path counting measure we have to look an average of ≈ 145 steps into the future.

3.1.2 Comparison of the two approaches

In the previous sections we have described two approaches to applying the principle of FSM to systems with small, discrete state spaces. The first of these is the path maximisation approach (most similar to the causal entropic forces framework), and the second is the end point maximisation approach (more similar to the empowerment framework). One might expect that making decisions so as to maximise either of these quantities would, under many circumstances, result in qualitatively similar behaviour. This is because having access to many different possible end states would usually require having access to many future paths, and vice versa. Equivalently, we can think of this as saying something along the lines of having control over the possible end points you have access to will generally require that you also have a large amount of control over all of the intermediate points too. We have shown this intuition to have some validity in certain situations for the two simple systems which we have studied, however there are also some differences. A clear advantage of the end point maximisation approach is that the number of steps into the future which need to be modelled before you see the expected interesting behaviour is typically significantly lower than with the path maximisation

approach. The main disadvantage is that you have to be more careful with the selection of the future path length n , since making it too large can lead to all of the states looking identical under this measure. This never happens with the path counting maximisation, which allows for n to be made arbitrarily large without altering the observed behaviour. However, when it comes to more complex situations we are almost always limited by n . As such, practically it seems like end point maximisation should often be preferable, or in other words, empowerment over causal entropic forces, at least if we are following a local gradient.

3.1.3 Extension to discrete Markov chains

Instead of just considering a system that is described by a graph where transitions occur deterministically between nodes, we now consider using a *probabilistic* model of how the system will behave in the future. Specifically, in this section we will consider only systems with a discrete number of states where the model of the future evolution can be described as a Markov chain, i.e. that the probability of transitioning to a particular state depends only on the state that the system is currently in. In this case, rather than an adjacency matrix we can instead define a “transition matrix” P . This fully describes the model of the future that we are using, such that $P_{i,j}$ is the probability of going from state i to state j in the next time step. For a system that can exist in N states, this is again an $N \times N$ matrix.

Considering future paths of length n , we can write an expression for the causal path entropy of these systems in terms of a sum over the path probabilities \mathbb{P}_{path} :

$$S_C^{(n)} = \sum_{\{\text{n-step paths}\}} \mathbb{P}_{path} \log(\mathbb{P}_{path}) \quad (3.7)$$

However, since the model for the future evolution of the system is Markovian we can write the probability of any particular path that goes through the sequence of states $\{i_0, i_1, i_2, \dots, i_n\}$ in terms of the transition matrix in the following way:

$$\mathbb{P}_{path} = P_{i_0, i_1} P_{i_1, i_2} \dots P_{i_{n-1}, i_n} \quad (3.8)$$

Hence the causal entropy associated with any particular state i_0 can be written as:

$$S_C^{(n)}(i_0) = - \sum_{\{i_1, \dots, i_n\}=1}^N P_{i_0, i_1} P_{i_1, i_2} \dots P_{i_{n-1}, i_n} [\log(P_{i_0, i_1}) + \log(P_{i_1, i_2}) + \dots + \log(P_{i_{n-1}, i_n})] \quad (3.9)$$

If we define a matrix $P \log P$ such that $[P \log P]_{i,j} = P_{i,j} \log(P_{i,j})$, we can define the “causal entropic matrix” as follows:

$$S_C^{(n)} = - \sum_{t=1}^n P^{t-1} (P \log P) P^{n-t} \quad (3.10)$$

such that the causal entropy of any state i_0 can be written as:

$$S_C^{(n)}(i_0) = \sum_{j=1}^N [S_C^{(n)}]_{i_0, j} \quad (3.11)$$

i.e. the sum of the row i_0 of the causal entropic matrix. The procedure to calculate $S_C^{(n)}$ is clearly very similar to the method of counting paths with the adjacency matrix introduced in the section 3.1, however here we have to evaluate the sum of n matrix multiplications rather than just one.

We can use a similar approach to calculate an entropy over the the accessible future states n time steps into the future. To do this, we just need to make use of the transition matrix P , as $(P^n)_{i,j}$ gives you the probability of being in state j after n -steps, given that you started in state i . As such we can define the entropy matrix $S^{(n)} = P^n \log(P^n)$ so that the entropy over the possible states accessible n -steps into the future (given that the system is currently in state i_0) is given by:

$$S^{(n)}(i_0) = - \sum_{j=1}^N (P^n)_{i_0, j} \log [(P^n)_{i_0, j}] \quad (3.12)$$

For either choice here of S_C or S we can then follow a gradient in this value to find a local maximum.

We can make one further extension here by considering a situation where we have actions available that do not move us deterministically between states. If we define an $1 \times N$ row vector $\langle a_i(i_0) |$ defined as the vector of probabilities of being in each possible state at the next time step, given an take action a_i from state i_0 , then the future entropy

associated with taking action a_i when in state i_0 is given by the sum of the vector:

$$\langle a_i(i_0) | S_C^{(n)} \text{ or } \langle a_i(i_0) | S^{(n)} \quad (3.13)$$

i.e. just the weighted sum of the probability that the action a_i takes you to a particular state multiplied by the future path entropy from that new state.

In both of these cases it is worth discussing exactly how this differs from the empowerment framework. Essentially, what we do is to choose the action in the present that moves us towards state(s) that have the largest future “freedom” (measured either as the entropy over future states or future paths). However, to do this requires us to be able to model the future evolution of the system out to some future time horizon. This means that we have a default model for the future evolution of the system, i.e. a model which does not account for the fact the system will actually be making decisions based on FSM. Note that this is also the case with the causal entropic forces framework; the path integral is taken over some probabilistic model we have for the system’s future evolution. The full empowerment framework does not have this issue because it explicitly looks at how much influence the *sequence of actions* $\{a_0, \dots, a_n\}$ has over the future sensor states after n time steps. However as was mentioned, this is one of the reasons it is a very difficult quantity to calculate, especially for large n . When constructing a transition matrix to model the future evolution of the system the simplest assumption is that the agent’s future actions will be chosen uniformly at random from the set of actions that are available in each state. Doing this certainly gives a reasonable measure of the amount of things that *can possibly* happen, despite not accounting for the fact that actual future actions will be chosen so as to try and maximise future freedom. However, ideally one would like a way to prune future actions which are extremely unlikely, such that they don’t need to be considered. One possibility could be to calculate an initial transition matrix P based on the first assumption (uniform future actions), and then use this to model the future actions in the construction of a new transition matrix for the modelled future behaviour. This could potentially be iterated multiple times.

3.2 Future state difference comparison

The methods in the previous section are good for studying toy problems with relatively small discrete state spaces, however when we look towards more complex systems of interest they are not really practical. The empowerment framework introduced in section

2.2 is also difficult to apply to complicated systems, since a full calculation requires (at least) enumerating all possible action sequences that the agent has available out to some future time horizon. Dealing with large or continuous state or action spaces can also lead to other issues. As an example of this, let us consider a circular agent in two dimensions equipped with some simple visual sensors (which will be used in Chapter 4 as the basis for our model of collective motion). If we imagine dividing the agent’s angular field of view into a discrete number of sensors which measure the geometric projection of other surrounding agents. A sensor reads a 1 if it is more than half full, and is zero otherwise. This is shown in figure 3.6 for two different sensor numbers. In a deterministic environment, calculating the empowerment involves directly enumerating the number of *distinct* visual states that are accessible using the available action sequences for however far into the future we are considering. If we have only a small number of fixed available actions, then as we increase the number of sensors the agent has it becomes clear that eventually almost every visual state which is encountered will be distinct. This will result in an essentially flat empowerment landscape, with nothing to distinguish between states.

In the limit that the number of sensors goes to infinity, in principle there should be a way of making the set of actions continuous (even just by adding some noise) that can allow for a sensible definition of empowerment to be retained, however doing so is not trivial. As such, it would be desirable to have a measure of the number of accessible future states that can more easily be generalised to a continuous state space. Furthermore, it would also be useful if this quantity could be estimated from Monte Carlo samples of future trajectories, rather than requiring a full explicit enumeration. In this section we discuss a possible candidate for such a quantity.

The key idea is simply to define a measure which compares how different any two given states of a system are. That is, for any two states \mathbf{x}_1 and \mathbf{x}_2 , we need to be able to define a function $d(\mathbf{x}_1, \mathbf{x}_2)$ that quantifies how different they are (for simplicity we will normalize this value so that it takes values between 0 and 1). In general there are a number of possible choices for this function, however there are many scenarios where one can easily make an intuitively sensible choice. From any given state (or for each particular choice of initial action), one can then simulate a number N of possible future trajectories under some model for the system’s future evolution, giving a set of final states $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$. To get an idea of the variety of possible states that are

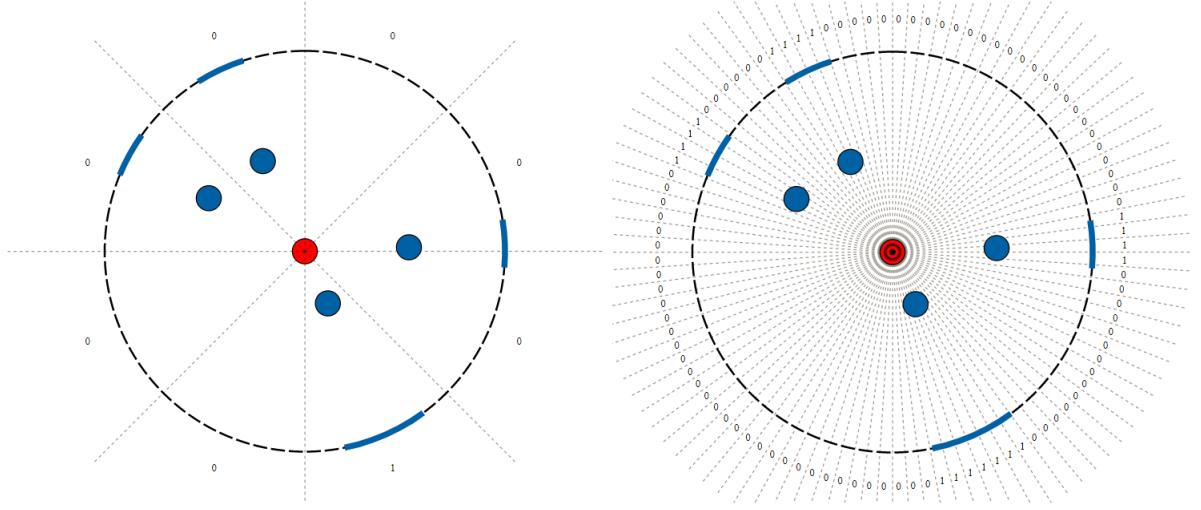


Figure 3.6: The effect of a large number of sensors on empowerment. Here the agent in red has its visual field of view divided up into a number of evenly spaced sensors. We then take the projection of the other agents surrounding it onto each sensor, which reads a 1 if more than half full, or 0 otherwise. On the left we show 8 sensors, whereas on the right we have 200. If we count the number of distinct states for a discrete number of short action sequences, then as the number of sensors increases it becomes more and more likely that every visual state encountered will be distinct. This means that the empowerment of every state will be almost constant.

accessible we then compute the average pairwise difference between them, that is:

$$\langle d(\mathbf{x}, \mathbf{x}') \rangle = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N d(\mathbf{x}_i, \mathbf{x}_j) = \frac{2}{N(N-1)} \sum_{i=1}^N \sum_{j=i+1}^N d(\mathbf{x}_i, \mathbf{x}_j) \quad (3.14)$$

(since $d(\mathbf{x}_i, \mathbf{x}_i) = 0$ by definition and $d(\mathbf{x}_i, \mathbf{x}_j) = d(\mathbf{x}_j, \mathbf{x}_i)$). There is no deep theoretical justification for this quantity, other than to say that intuitively it will be large when there are a lot of future states available that are different from each other and low when all accessible states are similar, i.e. when very few options are available. One could calculate a probability distribution over $d(\mathbf{x}, \mathbf{x}')$ instead of just the average, and then aim to maximise the entropy over this distribution, e.g. maximising:

$$S(d(\mathbf{x}, \mathbf{x}')) = - \int_0^1 \mathcal{P}(d(\mathbf{x}, \mathbf{x}')) \log(\mathcal{P}(d(\mathbf{x}, \mathbf{x}'))) \quad (3.15)$$

This would reward situations where the system has access to many states which are similar as well as many states that are different from each other, which could be more

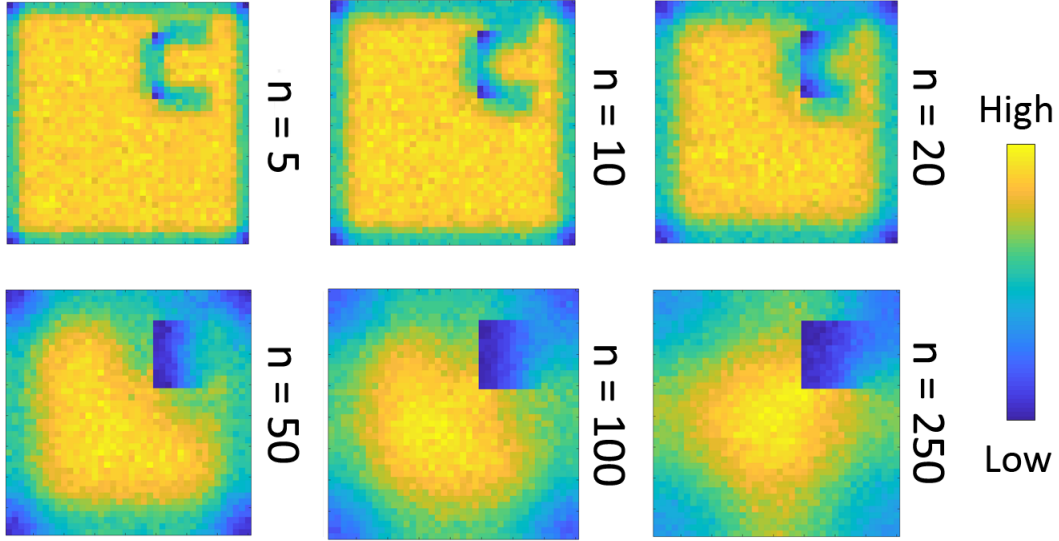


Figure 3.7: Heat maps showing the average difference in distance between end points for each initial starting point. Each point is calculated by simulating 1000 trajectories of length $n = 5$, selecting random moves. These can be compared to the analogous figures for the path counting and endpoint counting, shown in figures 3.2 and 3.3 respectively.

desirable in certain situations. However, this would be significantly more difficult to calculate, so we work with equation 3.14.

3.2.1 Application to a toy maze

To see how this measure compares with the empowerment and the future path entropy we consider applying it to the same system studied in section 3.1 (shown in figure 3.1). Here we run N_t random trajectories for n time steps into the future, starting from each possible initial position. We then look at the average difference between the end states that are reached (using equation 3.14). To do this we first need to define a way of comparing how different any two states are. The simplest way of doing this is to use the Euclidean distance between states (one can also use the shortest available path between the states and the results are essentially identical). That is, we say that $\mathbf{x}_i = (x_i, y_i)$ where x_i and y_i are the x and y coordinates on the lattice and define the difference to be $d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} / \sqrt{3200}$. Here the factor of $\sqrt{3200}$ is the maximum distance apart two states can be and just normalises the difference measure to lie between 0 and 1.

Figure 3.7 shows heat maps similar to those shown in figures 3.2 and 3.3, using $N_t = 1000$ and six different values of n . We see that this new measure is noisier (as expected, given that we are sampling a number of random future trajectories rather than exhaustively enumerating all of them), but nevertheless gives qualitatively similar results. That is, for large enough n it produces a field with an (approximate) gradient that would drive the system towards the middle, away from the restriction of the walls, where intuitively the freedom is maximised. It also shares the operating feature that the behaviour becomes approximately constant for sufficiently large n .

3.2.2 Application to a simple game

In this section we apply the future state difference measure to a more complicated situation which requires planning over a significantly longer time scale. We consider an agent occupying a simple grid world with a number of stationary blocks placed randomly. Initially these are unmoveable, however there are also a set of “gloves” located in the bottom right-hand corner which if collected allow for the blocks to be pushed and pulled into unoccupied spaces. After 50 time steps, the grid starts to fill with “lava”, which starts at the edges and moves in to any neighbouring, unoccupied square at the next time step. If the lava touches the agent it dies and the game is lost, however by manipulating the blocks it is possible for the agent to build a shelter to protect itself (see figure 3.8(e)). If the agent is still alive after the lava has spread across the whole grid then the game is won.

Although slightly contrived, developing an algorithm to win this game without a human providing task-specific knowledge is difficult. One obvious approach might be to try a reinforcement learning algorithm, providing a positive reward if the agent survives and negative reward if it dies, and then leaving the system to try to establish how to consistently obtain the reward by itself. This, however, runs into problems due to the sparsity of the reward. Reinforcement learning algorithms work by balancing exploration (trying out new actions in a given situation to see how they do) with exploitation of the current knowledge it has about how best to maximise the reward. To improve it has to be able to explore and discover which actions work well, however in this situation this is difficult since random sequences of actions will almost never obtain the reward. This is because the agent would have to randomly pick up the gloves, bring all of the blocks together to build a shelter, and then wait there until the lava passes. This requires a large number of very specific actions to be taken sequentially (in time), making it extremely unlikely.

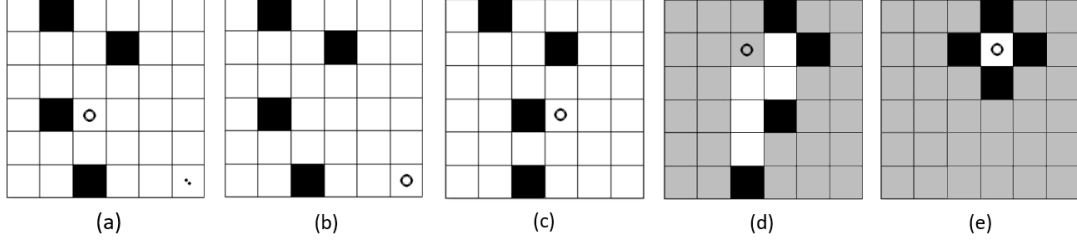


Figure 3.8: The agent (circle) starts in a grid-world with four initially unmoveable blocks and a pair of gloves in the bottom right-hand corner. If the agent picks up the gloves it is then able to both push and pull the blocks to rearrange them. At a later time “lava” starts to pour in from the sides and fills squares connected by an edge. The blocks are able to stop this. However if the lava touches the agent (as in (d)) then it dies and loses. If it is instead able to build a shelter for itself (as in (e)) then it survives and wins.

A standard reinforcement learning algorithm will require the positive reward to be discovered many times in order to learn, and so taking this approach is hopeless as there simply aren’t enough successful examples to learn from. By applying FSM the idea is that the agent will naturally be drawn to pick up the gloves, as this allows the blocks to be manipulated into many different potential states. It should then also be encouraged to keep as many blocks close to it as possible since this gives it access to the widest range of possible future states, with the absolutely optimal state being surrounded in a protective shelter.

To apply the future state difference comparison we first need a way of quantifying the difference between any two states that the system can be in. Whilst there are a number of ways this could be done, to demonstrate how this technique can be applied to much higher dimensional problems than the empowerment framework with relative ease we consider a full pixel-wise difference of the two states. That is, given two states \mathbf{x}_1 and \mathbf{x}_2 which can be represented with images similar to those in figure 3.8, we define their difference to be the average difference between the pixels in each image:

$$d(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{255N_{pixels}} \sum_{pixels,i} |v_1(i) - v_2(i)| \quad (3.16)$$

where $v_1(i)$ is the value of pixel i (between 0 and 255 as the images are grayscale) in state 1 and $v_2(i)$ is the value in state 2. We can simplify this as the system is made up of 36 squares which can be in one of only five states (empty, occupied by the agent, occupied by a block, occupied by lava or occupied by the gloves), and hence we can simply cal-

calculate the difference between each possible pair of squares and sum over these instead. We then choose a number of time steps τ to model future trajectories for, as well as the number of trajectories N so that we can use equation 3.14 to calculate the average difference between possible future states. The difficulty with this system is that building a shelter in general takes a lot of time steps, and whilst the state which has the highest average future state difference is the one in which the agent is surrounded by blocks on all sides getting to this usually involves passing through a number of intermediate steps which all have lower values. An example solution from a particular random initial configuration is shown in figure 3.9. Although this solution is probably not optimal in terms of minimising the number of steps to build a shelter any improvement would be relatively small, and we see that it takes a total of 37 different actions to complete. Also if we look at the fifth panel where the agent is surrounded by three blocks but where the fourth one is still quite far away, it is clear that in order to move this final block into place it is necessary to pass through a number of states where the average future state difference is going to be lower than the current state. This means that simply following a local gradient in average future state difference is not going to work well here. Potentially this could be overcome by making τ , the number of modelled future steps, very large, but this would require that we also make N significantly larger as well to ensure that the variance of the estimate given by equation 3.14 is small enough to give a reasonable estimate.

Here the approach we take is to instead consider only a small value of $\tau = 4$ (and $N = 200$) to calculate the actual value of any given state, but then to explore using much longer random trajectories. Along each of these random trajectories we then evaluate the value of each state which is encountered (using $\tau = 4$), looking for the highest value state anywhere along any of the random sequences we sample. That is, from a given starting point we simulate a number N_r of trajectories of length τ_2 , where each move is chosen at random. At each state along each of these trajectories we then evaluate the value using equation 3.14 with $\tau = 4$ and $N = 200$. We then choose the state (which does not have to be at the end of a trajectory) with the highest value and update the system by taking the sequence of actions which led to this state. This helps with exploration, preventing the system from getting stuck in a local maxima of the average future state difference (with $\tau = 4$). This does, however, come at a cost - this procedure is very computationally expensive. This is because, at each state of each of the N_r random trajectories, we have to generate $N = 200$ additional trajectories, each of length 4, to estimate the state's value. On the positive side this is extremely easy to

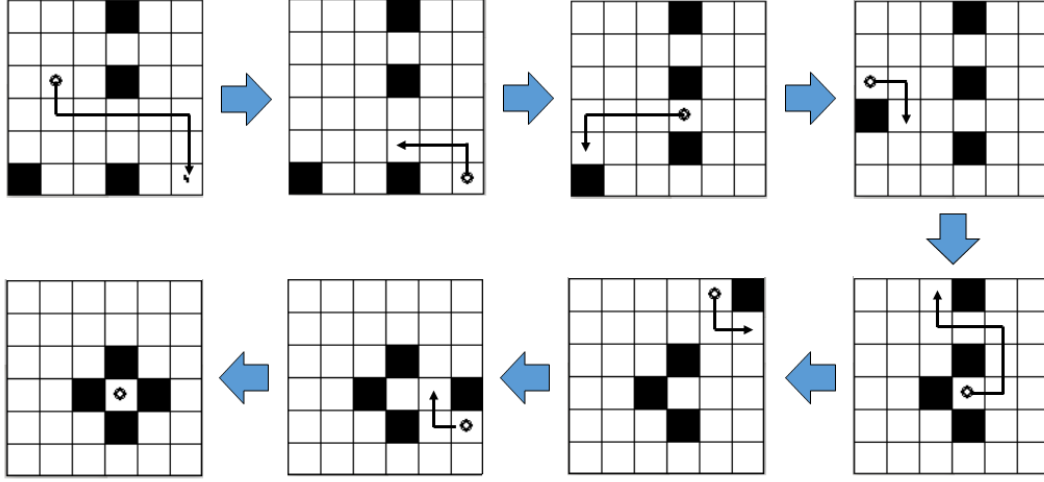


Figure 3.9: One way of creating a shelter from a particular random set up. In the first panel the agent collects the gloves located in the bottom-right square. From panels 2 to 3 the agent pulls one of the blocks up one square before collecting another block. The whole process takes a total of 37 steps to complete (it can most probably be done in less time than this, but not significantly so).

parallelise, but to avoid having to do this whilst still being able to run numerous long simulations we instead decided to train a neural network to learn to estimate the value function for an arbitrary state, when $\tau = 4$ and $N = 200$. To do this, we describe a state with a vector of input features which includes the agent's absolute position (x and y coordinates), whether or not the agent has the gloves and then the relative positions of all of the blocks. We then generate 5 million random situations where the blocks and agents are all placed, choosing some fraction where the agent has the gloves (90%) and some fraction (10%) where it doesn't. For each of these states we calculate a (stochastic) average future state difference, using equation 3.14 with $\tau = 4$ and $N = 200$. These results are then used as the targets to train the neural network. Details of the neural network architecture used and the training method employed are shown in figure 3.10.

Whilst this trained neural network does not perfectly reproduce the full value function, it nevertheless produces a function which is virtually identical (at least qualitatively), and which is maximised when the agent is surrounded by blocks in a central position in the grid. Using this we can run simulations where we choose large values for both N_r and τ_2 to test whether the system is able to plan enough steps into the future to be able to build a shelter and survive. As a demonstrative example, we choose $\tau_2 = 15$

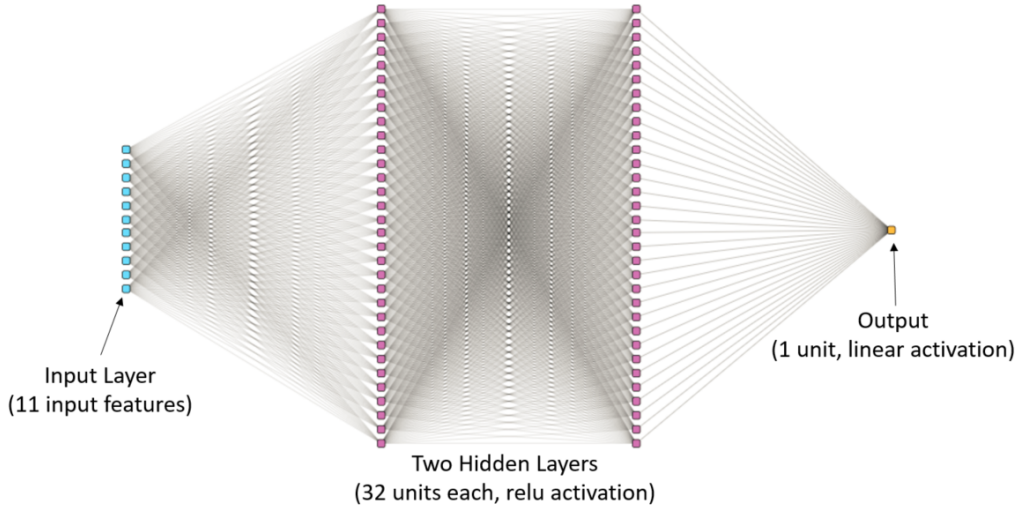


Figure 3.10: Details for training the neural network. The input layer takes the x and y coordinates of the agent, the relative positions of the four blocks and whether or not the agent has the gloves. The target output is then the calculated average future state difference from equation 3.14. There are two hidden fully connected layers of 32 units each, both using rectified linear unit activation functions[48]. Training is carried out on 5 million samples aiming to minimise the mean-squared error between the predictions and the targets using the ADAM stochastic optimisation routine[49] for 100 epochs, with batch sizes of 64. This was all implemented using the Keras library[50].

and $N_r = 20,000$ and run this for 100 different random initial configurations. In 78/100 of these trials the agent was able to successfully build itself a shelter and survive. A video showing some of the successes, as well as a few of failures, starting from different random initial configurations is included as SI movie 1. We see that even in a number of the cases which fail the agent gets close and given a bit more time until the lava enters would most likely succeed. In a couple of cases the agent gets trapped with only three blocks and is unable to get over this “potential barrier” to bring the fourth one in. This usually happens if three of the blocks are close to one of the edges and the fourth block is on the opposite side (although even in some cases when this happens the agent is still able to successfully build a shelter).

Ultimately, this example shows how the relatively sophisticated behaviour required to solve a puzzle can be spontaneously induced from a simple task-independent motivational principle. It is also a simple demonstration of how the effective time horizon of

the FSM principle can be extended by supplementing the explicit calculation over τ future time steps (where τ is much less than the time horizon we need to plan over) with a random search, rather than following a local gradient, to prevent the system from getting stuck in local maxima.

Chapter 4

Intrinsically motivated collective motion

In this chapter we present our primary application of the principle of FSM. We study a group of agents, each equipped with simple visual sensors, which make decisions about how to move so as to maximise the amount of control they have over the visual states that they have access to in the future. As we shall see, making their decisions solely based on this principle spontaneously leads to the emergence of a highly ordered, cohesive and robust swarm with rich collective dynamics. In contrast to the models discussed in Chapter 2, this occurs without the need for any explicit co-alignment or cohesive interactions, arising purely as a result of each agent individually trying to maximise the control it has over its (visual) future. The primary motivation here is not to come up with a realistic model of any particular animal system, but rather to analyse a non-empirical, “bottom up” approach to collective motion based on a simple principle (FSM). This could then potentially represent a useful conceptual framework for understanding the collective behaviour of real organisms which are capable of seeing the world around them and processing information. Despite this, we shall see that there are a number of striking similarities between the results obtained from simulating this model and data obtained from observing real starling murmurations[37].

4.1 Description of the base model

We consider a group of N finite-sized, circular agents of radius R moving with nominal speed v_0 on an infinite 2D plane. Each agent has a discrete number of visual sensors, n_s , which act like simple retinas to detect a geometric projection of the other agents around

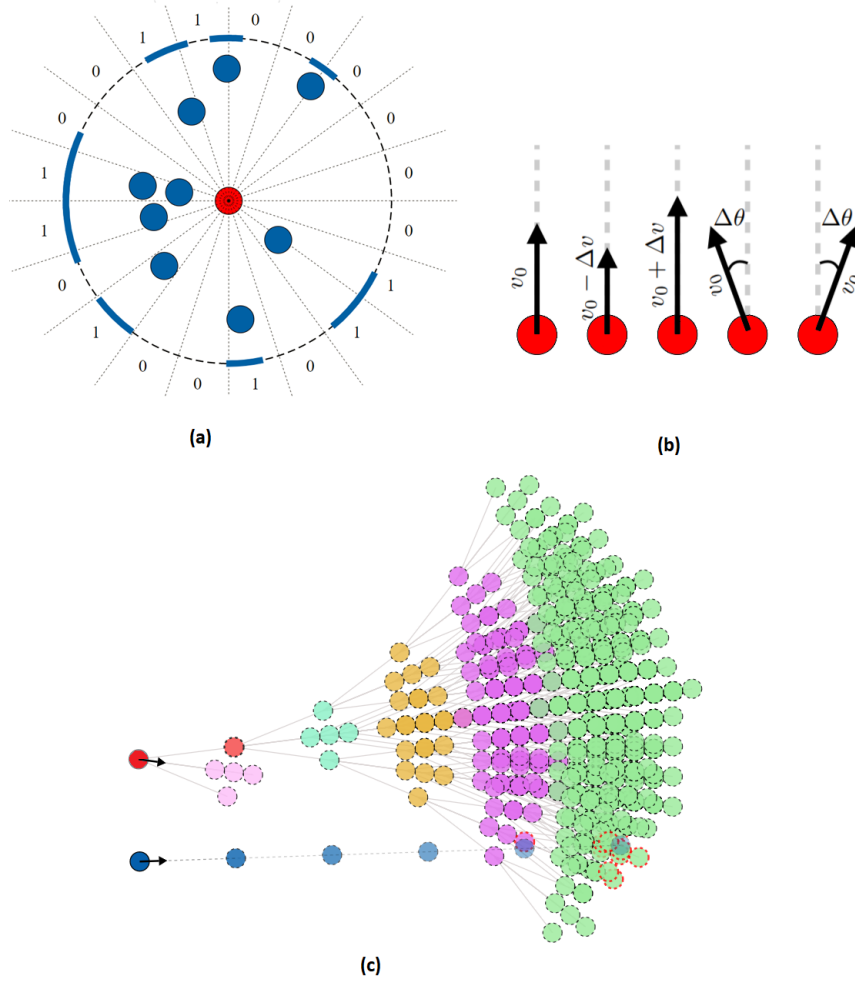


Figure 4.1: (a) Definition of the visual state of an agent. The agent's (red) angular field of view is divided up into a finite number of sensors and the geometric projection of the other agents is taken (represented by the solid blue arcs). If a sensor is more than half full it reads 1, otherwise it reads 0. As such, the visual state of each agent can be represented as a Boolean vector. (b) Each agent has 5 possible actions available at each time step: continue along its current direction with speeds v_0 , $v_0 + \Delta v$ or $v_0 - \Delta v$, or change its orientation by an amount $\pm\Delta\theta$. (c) To choose how to update itself each agent performs a “tree search” of its possible future states for each of the five currently available moves. When modelling these hypothetical future trajectories, the decision-making agent assumes that the other agents (blue) will continue to move in a straight line. Each node on this future tree will have a visual state associated with it and the agent chooses whichever of the moves in the present leads to the future tree with the *largest number of distinct visual states* contained within it. Branches where a collision is predicted (those with red outlines) are cut off and not included in this count. Here $\tau = 5$, and the future tree is shown for the move of reorienting by an amount $\Delta\theta$ to the left (red).

it (see figure 4.1(a)). Each sensor either reads 1 if the fraction of it which is filled is more than s_{tol} , which we take to be equal to a half unless otherwise stated, or a 0 otherwise. At each time step every agent has a choice of five possible actions which allow it to change its current orientation or its speed, as shown in figure 4.1(b). Keeping the number of actions and the number of sensor states discrete in this way allows us to directly apply the empowerment framework outlined in Section 2.2, although later on in the chapter we will think about ways of modifying this model to deal with continuous visual states. To make each decision the agents construct a tree of all of their potential future trajectories out to some time horizon τ . On each of the hypothetical nodes contained within this tree they then calculate the visual state that they would encounter were they in this position at a future time — contingent on a model for how they anticipate that the other agents are going to move. This requires making a prediction for how they expect the other agents to move during these hypothetical futures. Later in the chapter we shall explore different choices for this. For the time being we adopt the simple option of modelling the other agents as moving in a straight line following their current velocity at nominal speed v_0 . The decision-making agent then counts the number of *distinct visual states* associated with each initial branch of this tree, which correspond to each of the moves which are available in the present, and chooses whichever one maximises this number. A visualisation of this process is shown in figure 4.1(c). This amounts to an almost direct application of the empowerment framework since we are dealing with a deterministic environment, with the slight difference that we count all distinct states on the whole tree rather than only those at the final future time step. Note that branches where collisions are predicted to occur are truncated and do not contribute to the distinct visual state count. Consequently, collisions will only occur if they are unavoidable given the available actions or if the predictive model for the future movement of the other agents is not fully accurate. Every agent carries out the same procedure, each making the same assumptions about how the other agents are going to move, to make its decision *for the next time step only*. This makes the algorithm rather computationally demanding, especially as τ is increased, however the essence of the procedure is extremely simple; agents simply move to maximise the number of distinct future visual states that they anticipate having access to.

There are a number of parameters in this model which need to be set in order for it to be simulated. For some of them there is not necessarily an obvious criteria for how they should be chosen, and so this warrants some discussion. For simplicity we can set the time and length scales by choosing the time step duration Δt and the agent radius

R to be equal to 1. We then have to decide on the values for number of sensors n_s , the default speed v_0 , the reorientation angle $\Delta\theta$, the difference between allowable speeds Δv and τ , the number of time steps into the future that we model hypothetical trajectories for when making a decision. The parameters which we have the most freedom to choose here are n_s and τ . In figure 3.6 we discussed how making n_s too large would lead to an issue where virtually every accessible state is distinct, leaving the agent in a position with nothing to determine between moves except for predicted future collisions — that is, vision would be playing no active role and the agents would just separate from each other (it’s worth noting that in the continuous version of this model, presented later in the chapter, this can never occur and so is not a problem). The same issue can occur if τ is too small, even for a reasonable choice of n_s . If n_s is too low (and potentially if τ is made too large), it is then possible for each branch to explore every possible visual state, again leading to a situation where there is no mechanism to distinguish between moves. Another reason why making τ too large could be undesirable is that the agents are using an imperfect heuristic to model the motion of the other agents in the hypothetical future trajectories they consider. Generally this will get less accurate with increasing time steps. This would mean that what the group of agents actually does is substantially and qualitatively different from what each individual expects the group to do. Taking these considerations into account we choose a nominal set of parameters to study as follows: $n_s = 40$, $\tau = 4$, $\Delta v = 2$, $\Delta\theta = 15^\circ$ and $v_0 = 10$. Whilst there is no deep reason for these choices they work well and we shall find that the model is remarkably robust, producing qualitatively similar behaviour over a wide range of parameter values.

4.2 Simulations of the base model: collective motion

To simulate the model we initialise the agents randomly into a box of a given size and align them along the x-axis, adding a small amount of Gaussian noise to their initial orientations (typically with a standard deviation between $\Delta\theta$ and $2\Delta\theta$). We choose to start the agents in a state which is of relatively high order, because otherwise they tend to break up into smaller sub-groups which generally will not later amalgamate. This occurs because we are using modest values of τ , and so if a sub-group forms where the agents within have future trees which strongly overlap, then they are much more strongly influenced by the motion of the agents within their group compared to the agents in other sub-groups. Whilst we cannot probe very large values of τ due to the computational cost, in figure 4.2 we demonstrate that the swarms become more robust to the initial conditions we choose as τ is increased, at least for τ up to 6. Initially we

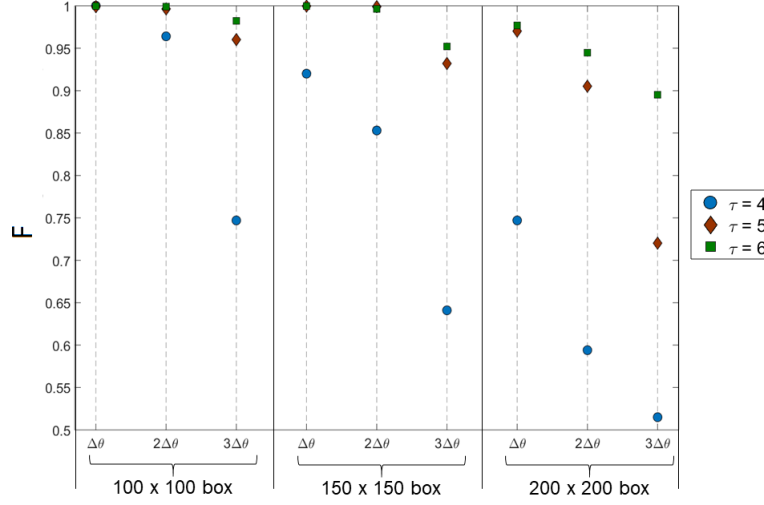


Figure 4.2: Simple demonstration showing how the robustness to the initial conditions in which the swarm is set up improves with increasing τ . Here we consider starting the agents positioned randomly within boxes of sizes 100×100 , 150×150 and 200×200 with different amounts of noise added to each agent’s initial orientation (standard deviation equal to $\Delta\theta$, $2\Delta\theta$ and $3\Delta\theta$), for $\tau = 4, 5, 6$. We then measure the fraction of the agents which remain in the swarm after 100 time steps, F , averaged over 20 different simulations. Whilst the choice of 100 time steps is quite arbitrary, usually the swarm has equilibrated by this point (at least for $N = 50$). Details of exactly how we define whether an agent is in the main swarm or not are described later on in the text. In every case we see more agents are left for larger values of τ .

confine our attention to small swarms, choosing a nominal value of $N = 50$, primarily for computational reasons — although later in the chapter we shall investigate larger swarms of up to $N = 500$ agents which we are able to obtain by parallelising the simulations to be ran on many cores at once.

The primary result we find is that the simulations reliably lead to highly ordered, robust swarms which remain cohesive and naturally regulate their density. A typical snapshot of a swarm and the trajectories that the agents follow using the nominal set of parameter values is shown in figure 4.3(a), at two different time steps. A video of this simulation is also included in the supplementary materials as SI movie 2.

Before going on to measure various quantities of interest we need a way of clearly defining what constitutes the “main swarm”. This is because we are working in infinite 2D

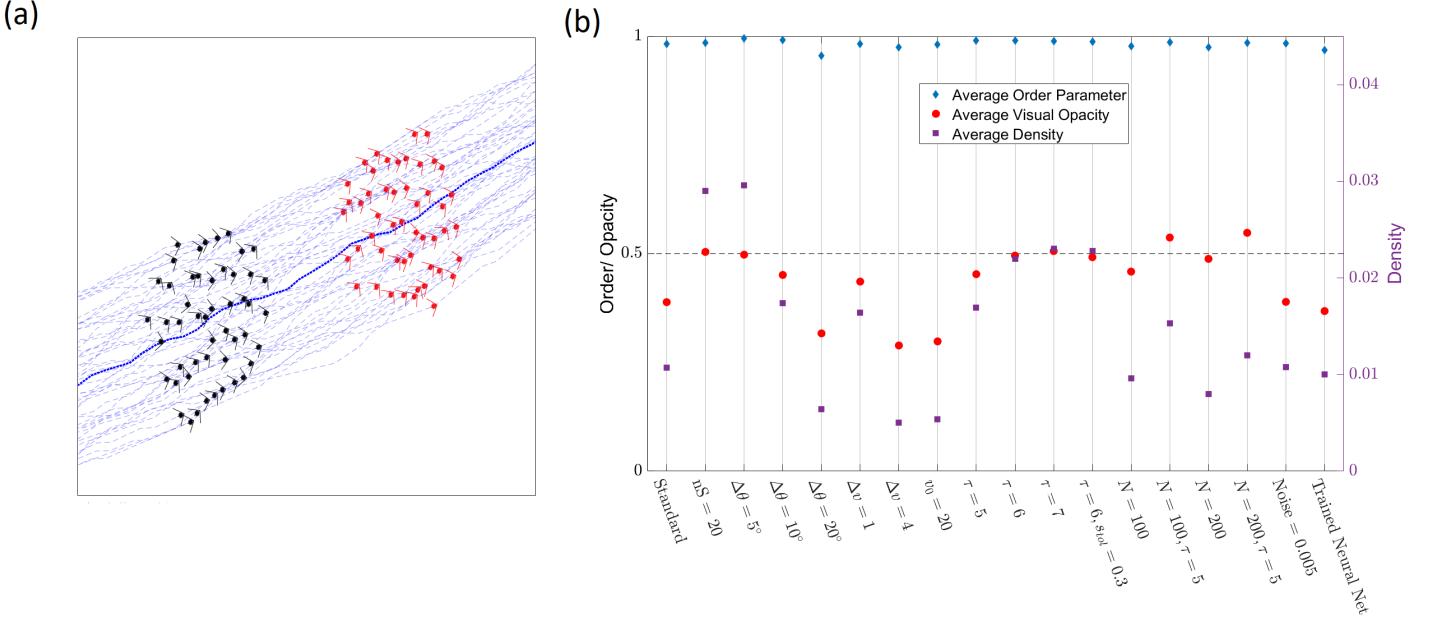


Figure 4.3: (a) Snapshots of a swarm where agents are moving according to the described FSM algorithm, shown at two different times. The thick blue line shows the trajectory that the centre of mass follows and the lighter blue lines show the individual trajectories. The simulation employs the nominal set of parameters ($n_s = 40$, $\tau = 4$, $\Delta v = 2$, $\Delta \theta = 15^\circ$, $v_0 = 10$, $R = \Delta t = 1$), and a video is included in the supplementary material (SI movie 2). (b) A plot showing the average value of the swarm’s order, opacity and density for a range of different parameter values. The “standard” values are the nominal values mentioned above, and the labels identify those parameters that are varied. All averages are calculated over 1000 time steps having let the swarm first equilibrate. We see that for all of these parameter variations we retain a swarm that is highly ordered and close to marginal opacity.

space, meaning that it is possible for individuals or small sub-groups to fragment from the main group and never return. If this happens, we do not wish to account for the fragmented agents when calculating quantities such as the swarm’s order. As such, we define a swarm connectivity graph between agents at each time step, where we say that any two agents are “connected” if they are within a distance of $20R$ of each other (here R is the agent radius). This is an arbitrary choice and many other values could be chosen without significantly altering the results. For context, with $N = 50$ the average swarm size is around $70R$. We then define the “main swarm” to be the largest connected component of this graph[51]. Figure 4.3(b) shows a plot of three quantities of interest — the average order parameter, the average opacity and the average number density of

the resultant swarms, for a variety of different parameter choices. The order parameter ϕ is a measure of the overall swarm alignment and can be defined as follows:

$$\phi = \frac{1}{N} \left| \sum_{i=1}^N \hat{\mathbf{v}}_i \right| \quad (4.1)$$

where $\hat{\mathbf{v}}_i$ is the unit vector pointing in the direction that agent i is currently travelling in. Consequently, it takes a value between 0 and 1, with 1 corresponding to a perfectly ordered swarm. We see for all parameter variations considered the resultant swarms are extremely highly ordered, with values typically exceeding ~ 0.98 . The parameter which seems to have the largest effect on the order is $\Delta\theta$, with smaller values leading to even higher orders (although we note that if $\Delta\theta$ is made very small the swarms tend to become unstable, presumably because it takes many time steps to make significant adjustments to their relative configurations).

The number density ρ can be defined as $\rho = \frac{N}{A}$, where A is the area within the convex hull bounding the main swarm[52]. The average opacity is also shown and is defined simply as the average fraction of each agent's visual field which is filled by the projection of the other agents. We find that for all parameter values the opacity is close to 0.5, seemingly converging ever closer towards this value as τ is increased. This *marginal opacity*[46] is a feature that is observed in flocks of starlings. One might assume that it arises in our model only because of the (natural) choice of $s_{tol} = 0.5$, however we also include a simulation where $s_{tol} = 0.3$ and find that an average opacity close to 0.5 still emerges. Later in the chapter, we shall also introduce a more continuous measure of the degeneracy of future visual states which does not require dividing up the field of view into a discrete number of sensors. Here we also find that marginal opacity emerges, suggesting that it is a natural consequence of FSM.

We also calculate the correlation function of fluctuations around the swarm's instantaneous mean velocity. It has been suggested that many biological systems may be poised at criticality[53], and one characteristic supporting this, observed in starling murmurations,[37] is that they exhibit scale free correlations. This means that the correlation length of the system scales linearly with increasing size, such that there is no characteristic length scale over which individuals remain correlated. This is thought to, amongst other things, improve the quality of the swarm's collective response to external perturbations such as the detection of a predator. To study this we define the instan-

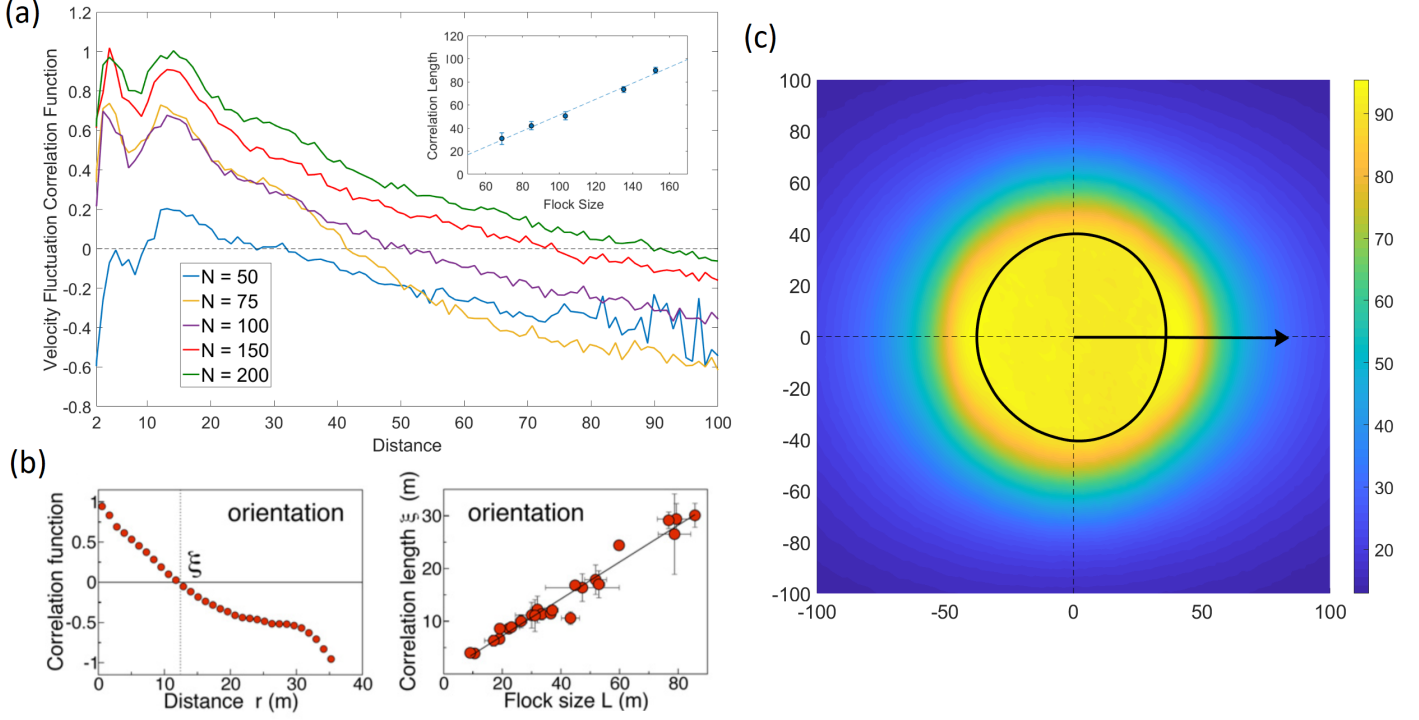


Figure 4.4: (a) Plot of the velocity fluctuation correlation functions we observe from simulating our model for a range of different swarm sizes (from $N = 50$ to $N = 200$). The inset figure shows the correlation length, defined as the point at which the correlation function crosses the x -axis (after its highest point) as a function of the physical size of the swarm. (b) Taken from Cavagna et al.[37]. Correlation functions and correlation lengths calculated from data obtained observing real murmurations of starlings. (c) A heat map showing the average number of hypothetical future visual states available to an agent given its position within the swarm — see in the main text for more details about how this is calculated. The x -axis is the average centre of mass velocity and the solid black line represents the average convex hull of the swarm. We see that within the swarm this measure is remarkably uniform, suggesting that agents are generally just as happy to be in the centre of the swarm as they are to be on the edges.

taneous velocity fluctuation of agent i as $\mathbf{u}_i = \mathbf{v}_i - \langle \mathbf{v} \rangle$, where \mathbf{v}_i is the agent's current velocity and $\langle \mathbf{v} \rangle$ is the average velocity of the main swarm. We can then look at how the average value of $\mathbf{u}_i \cdot \mathbf{u}_j$ between any two agents i and j varies as a function of the distance between them. This is shown in figure 4.4(a) for swarms made up of between 50 and 200 agents, normalised such that $\langle \mathbf{u}_i \cdot \mathbf{u}_j \rangle = 1$. Note that the plot only shows distances of more than $2R$ (with $R = 1$) since otherwise the agents will have collided. Figure 4.4(b) shows data from starling flocks taken by Cavagna et al.[37]. Whilst the behaviour of the correlation functions from our simulations at very short distances is surprising in that it has a sharp dip (particularly for $N = 50$), the overall similarity with the starling data is striking. Most interestingly we find that the correlation length, defined as the point that the correlation function crosses the axis (ignoring the initial dip for $N = 50$), also scales linearly with the size of swarm (defined to be the square root of the area of the convex hull), and hence that the correlations within this model are scale free. The similarities of our model with the starling data in terms of the observed order parameters, the marginal opacity and the scale free correlations is quite remarkable as we did not set out to empirically model the behaviour of starlings. Despite this, upon reflection, it makes some sense that a group of agents seeking to maximise the control they have over their future visual environment would choose to be in a state of marginal opacity since this is compatible with the largest variety of visual states. The scale free correlations also make sense as we would expect the swarm as a whole to prefer a configuration where collectively it most rapidly able to respond to external perturbations.

The final quantity we look at is the actual number of hypothetical future visual states that are available to an agent, and how this varies depending upon its position within the swarm (see figure 4.4(c)). To do this we run a standard simulation and at each time step calculate the current centre of mass and the centre of mass velocity. We then discretise space using a grid of points relative to the centre of mass, with grid points taking integer values between -100 and 100 in both the x and y directions, where the x -axis is defined by the swarm's current centre of mass velocity. At each of these grid points we place a "test agent" by moving whichever agent in the swarm is closest to the grid point under consideration and calculating the number of hypothetical visual states which would be available to it if it was there. We then replace the agent in its previous position and move on to the next grid point. This count is averaged over 10 different runs, each consisting of 1000 time steps. For guidance we include the black line which is the average shape of the swarm, defined in terms of its convex hull. To get this, at each time step we consider every angle $\theta \in [0, 2\pi)$ (discretised, but finely enough so as to

not be noticeable in the plot) and calculate the distance at which a radial line starting at the swarm’s centre of mass and oriented with angle θ intercepts the swarm’s convex hull. For each θ we average these values over the simulations to give us the average swarm shape shown in figure 4.4(c). Intuitively, one might think that an agent would have more hypothetical future visual states available to it if it starts somewhere nearer the middle of the swarm — however we see this is not the case. Within the typical size of the swarm the landscape of this measure is remarkably uniform, with no significant gradients on average. This suggests that generally an agent moving to maximise this quantity should be equally happy (on average) whether it finds itself in the middle of the swarm or out on the edges. This means that we should not expect there to be any significant flows present in the cluster.

4.3 Alternative models for the predicted trajectories of other agents in the swarm

A key ingredient in the set up presented so far is a model for how the decision making agent predicts the rest of the swarm will move within the hypothetical future trajectories considered. If we accept that FSM is a sensible principle for the agents to be following, then clearly it is preferable for them to be using the most accurate model possible when modelling the future motion of the other agents. So far, we have considered only a heuristic in which the other agents move ballistically in a straight line at speed v_0 . This is clearly only an approximation to the trajectories which will actually be realised, as can be seen from a quick glance at figure 4.3(a). Nevertheless, there is at least some level of self-consistency here because the resultant swarm has very high order and the trajectories do not deviate substantially from ballistic motion. Whilst it is an interesting result that we get collective motion using such a simple heuristic, we would like to develop heuristics which can be made more self-consistent with the FSM procedure.

To make the model fully self-consistent we need to be able to model the other agents as if they were also moving according to FSM. However, to do this they need a model for how the other agents are going to move themselves, which depends upon the current decision that is being made. This makes finding a closed solution which is completely self-consistent very difficult; potentially impossible. An alternative approach could be to build a hierarchy of models, in which at each level FSM is used contingent on the trajectories of the other agents being governed by the model at the level below it. So

for example, the first model could be to move according to FSM under the assumption that the other agents will move ballistically (our base model). Then, the next model up in the hierarchy would be to move according to FSM under the assumption that the other agents will move according to the first model (FSM with the ballistic assumption), and so on. By iterating this forward, one might hope that we could eventually converge towards a fully self-consistent solution — however this quickly becomes extremely computationally demanding. As such, in this section we focus on heuristics which are simple and computationally quick to calculate, but which we can tune in some way to make them at least partially self-consistent.

The first alternative we consider is a heuristic in which the other agents in the swarm

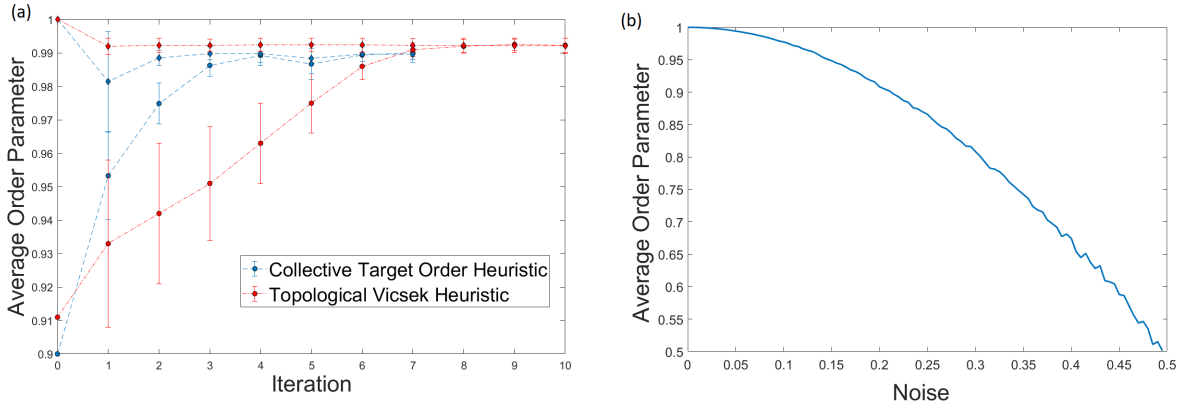


Figure 4.5: (a) Convergence towards a more self-consistent heuristic. We consider two heuristics which take as input an order parameter, and run the FSM procedure with these heuristics used to model the motion of the other agents in the hypothetical future trajectories. This produces a swarm with a new average order parameter, which we use as an input to the heuristic in the next iteration. Eventually we converge to a value where the output from the FSM procedure is consistent with the input order in the heuristic. A video showing the first three iterations using the “collective target order” heuristic is included in the supplementary information as SI movie 3. (b) The topological Vicsek model’s only control parameter is the noise added to each agent’s orientation at each time step. To convert this into an order we run simulations of the topological Vicsek model for many different noise values, each for 10,000 time steps. We then calculate the average order parameter, allowing us to map directly between a noise value to an average order, and vice-versa.

are updated in random order so as to collectively target a particular value of the order parameter: at each time step every agent, in a random order, chooses to either maintain

its current orientation or turn by an amount $\pm\Delta\theta$, based on whichever of these options will bring the *collective* order parameter of the swarm closest to its target value, ϕ_t . Whilst this is a somewhat unnatural procedure for the individual agents to be following, it nevertheless provides a simple way to generate a swarm with an order parameter which is very close to the specified input target ϕ_t . This is then a parameter which can be tuned in order to make this heuristic for the motion of the other agents as close to being self-consistent with the actual motion of the FSM procedure as possible. To implement this we start with a given target order, say $\phi_t = 0.9$, and run the FSM algorithm. Rather than assuming that the other agents will move ballistically in the hypothetical future trajectories, instead they are assumed to move to collectively target order ϕ_t , as just described. Running FSM with this heuristic will produce a swarm which has an average order parameter ϕ_r — which generally will not be equal to ϕ_t . However, once we have this we can re-run the FSM procedure, where now we use $\phi_t = \phi_r$, i.e. we target the order obtained from the previous iteration of the FSM algorithm. We can then repeat this procedure to find a target order which is consistent with the actual order that FSM produces under this heuristic. This is shown in figure 4.5(a) (in blue). In this we take two different initial starting states — one at $\phi_t = 1$, and the other at $\phi_t = 0.9$. The points plotted are the average order of the FSM procedure where we iteratively use the value of order realised at the previous iteration for the next ϕ_t . We see that indeed we are able to achieve convergence to an output order ϕ_r which is ultimately in excellent agreement with the input order ϕ_t . At this point, the agents are then using a heuristic for the motion of the others which is self-consistent at the level of the swarm’s average order (although of course the heuristic with this ϕ_t will still not perfectly mimic the FSM procedure). Note that if we start this iterative procedure with $\phi_t \lesssim 0.85$ then the resultant swarm is unstable and tends to fragment into smaller sub-groups — although these sub-groups still have higher order.

As a second alternative heuristic which is perhaps a more natural choice we consider the topological Vicsek model[43], mentioned towards the end of Chapter 2. This is a more natural choice for modelling the motion of the other agents as it is a well established model of collective motion and only involves interactions between neighbours, rather than requiring each individual to know about the swarm’s global order parameter. Here the agents update their orientations with a co-alignment interaction between their topologically assigned neighbours, subject to the addition of some scalar noise. The downside to this heuristic is that the only parameter we have to tune for better self-consistency with FSM is the noise, not the order parameter itself, and so running

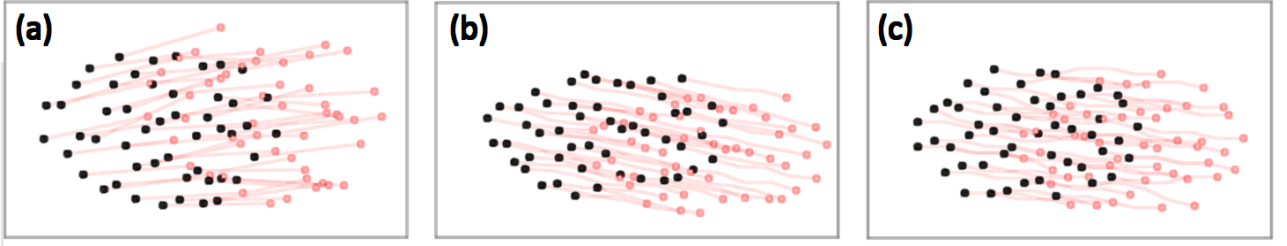


Figure 4.6: Snapshots of the swarms produced, as well as the modelled future trajectories for the three heuristics considered so far. (a) Ballistic heuristic: agents are modelled as moving in a straight line at v_0 . (b) The topological Vicsek heuristic. (c) The collective target order heuristic. Details for each of these are given in the text.

the FSM procedure does not directly give us a new noise value to compare with. This requires us to run simulations of the topological Vicsek model itself for a wide range of different noise values, as shown in figure 4.5(b). This information allows us to convert directly between the noise introduced into the topological Vicsek heuristic and an average order parameter. This can then be compared directly with the average order found from running the FSM procedure, and so we are able to proceed in the same spirit as we did with the previous heuristic, starting with a given value of the noise and iterating the FSM procedure forward to look for convergence. This is also shown in figure 4.5(a) (in red), and again we see that it converges to a self-consistent average order parameter.

To get a rough idea of how good these heuristics actually are at mimicking FSM we show some snapshots in figure 4.6 of the swarms that form, as well as the predicted future states out to $\tau = 4$. We see that over 4 time steps even the ballistic model is a reasonably accurate predictor, however there is a clear difference between the behaviour of the actual swarm and the predicted future trajectories which will quickly get worse if τ is made larger. For the other two cases (which are shown after their input orders have been tuned to be as self-consistent as possible), we see by eye that the predicted future swarm looks virtually identical to the actual swarm. Whilst we know that over a longer period of time these heuristics will get gradually worse (in part because they have no cohesion built into them), nevertheless this demonstrates how, over relatively small future time intervals it is possible to construct a simple heuristic which is more self-consistent with the FSM algorithm than the ballistic assumption.

4.4 Training a neural network to mimic the FSM procedure

We can generate a more realistic heuristic for the motion of the other agents by training a neural network to learn directly how to mimic the decisions made by the FSM algorithm. This could then be used to model the motion of the other agents over a much larger number of time steps whilst potentially still retaining its similarity to the actually realised FSM trajectories. Additionally, it would provide a useful demonstration of the plausibility of organisms being able to make decisions which may seem like they are using FSM, but without actually having to perform complicated calculations involving the consideration of many hypothetical future trajectories.

To do this we run simulations of the full FSM procedure and for each agent at each time step record the *current and previous visual state* that the agent sees (a $2n_s$ -dimensional vector), along with the decision it makes (an integer between 1 and 5). This means that we are only going to provide the neural network with *visual information that would be available in a real situation*, i.e. without the need to model any hypothetical future trajectories. It is interesting to note that the inclusion of the previous visual state, as well as the current visual state, is crucial in training a neural network which is able to qualitatively mimic the FSM algorithm. Once this data is gathered we essentially have a supervised learning problem[54]. That is, we have a set of inputs and a set of corresponding labels (the moves made by the FSM procedure), and we want to learn a function which is able to map between the two. Since we will never be able to gather enough data to build a look-up table telling us what the FSM algorithm would do for every conceivable input, it is necessary to use a method which can generalise well to unseen states. Since we are relatively unconstrained in terms of the amount of data we can gather (we can run as many simulations as we like, within reason), we decided to use a deep neural network[55] as a function approximator, since these are known to generalise well for a wide range of tasks when provided with sufficiently large amounts of data.

To gather the data we run the FSM algorithm with the “standard” set of parameters, under the assumption that other agents move according to the ballistic heuristic. We carry out 800 separate simulations, each for 250 time steps with an array of different starting configurations — agents are initially placed at random in a square box with dimensions that vary randomly between 80 and 160, and Gaussian noise with standard deviation $2\Delta\theta$ is added to their initial orientation. Including such a variety of initial con-

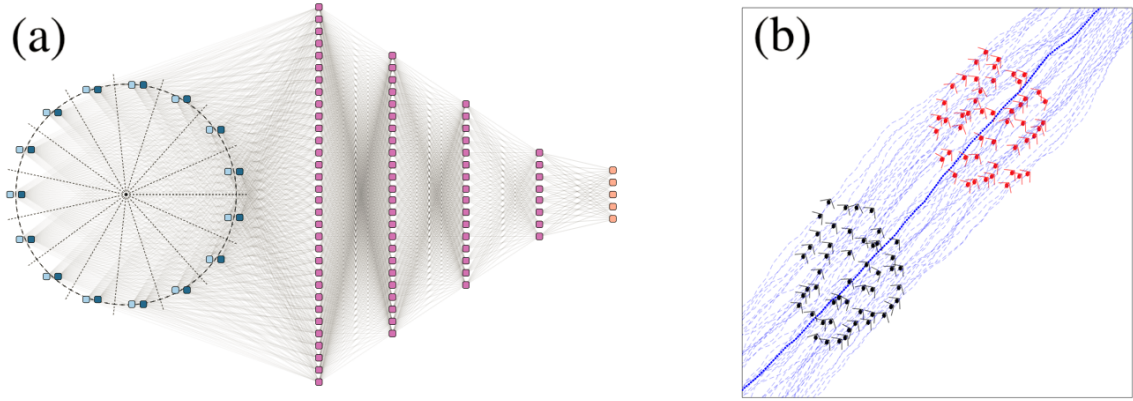


Figure 4.7: (a) The agent's current visual state, as well as the visual state at the previous time step, is provided as the input to the neural network. We provide the actual fraction of each sensor filled rather than just the 0 or a 1 that the FSM algorithm would use. The inputs corresponding to the previous visual state are shown in light blue, where as the dark blue inputs correspond to the agent's current visual state. These inputs are then fed through four hidden layers (of 200, 100, 50 and 25 units respectively) and attached to an output which gives an integer between 1 and 5, shown in orange. (b) Typical trajectories of the swarm produced when each agent moves according to the trained neural network. This can be compared to figure 4.3(a). A video showing a swarm moving according to the full FSM algorithm side-by-side with a swarm in which each agent moves according to the trained neural network is included in the supplementary materials (SI movie 4).

ditions provides the network with many examples to learn from that differ significantly from the steady state, and hence allows the network to learn to mimic FSM in these perturbed situations as well. We find that this significantly increases the robustness of the resultant swarms, allowing them to better recover from small fluctuations without fragmenting. In total, this gives us 10 million labelled examples to train on.

The neural network architecture we employ is sketched in figure 4.7(a) and consists of four fully-connected hidden layers containing 200, 100, 50 and 25 neurons respectively. The last of these is attached to a softmax classifier[56], outputting an integer between 1 and 5. Whilst it is usually difficult to gain any intuition for exactly how a deep neural network learns, the idea of starting with larger hidden layers which get progressively smaller as information is propagated forwards is that the earlier layers should be able to identify “lower-level” features of the input. These then feed into the subsequent, smaller layers which hopefully can learn “higher-level” features that are important in the decision making process. By having multiple hidden layers, the rationale is that this will allow a hierarchy of features to be learned at gradually higher and higher levels of abstraction. Despite this basic idea, we can make no real attempt to understand what these actual higher level features that the network might be learning are, and so the network can effectively be considered as a black-box function approximator. All of the hidden layer units use the non-linear ReLU[48] (rectified linear unit) activation function ($\text{ReLU}(x) = 0$ if $x < 0$, $\text{ReLU}(x) = x$ for $x \geq 0$). The network is trained on 90% of the data, leaving 10% aside for validation. This is carried out for 500 epochs using the ADAM optimizer[49] in Keras[50], with an initial learning rate of 0.0001.

Typical trajectories of a swarm produced using the trained neural network are shown in figure 4.7(b), and a video showing a side-by-side comparison with the full FSM algorithm is included in the supplementary materials as SI movie 4. We see that qualitatively they are extremely similar to the trajectories produced by the full FSM algorithm shown in figure 4.3(a). The average order, opacity and density for the trained network are also included in figure 4.3(b) and can be seen to be very similar to the full FSM algorithm, meaning that the agreement is both qualitative and quantitative. This is quite remarkable as the fully trained network only achieves a correct classification accuracy of $\approx 43.8\%$, i.e. over half of the time it is not able to correctly reproduce the exact move which would be made by the full FSM algorithm. This suggests that there may not be enough information contained in only the current and previous visual states to predict exactly what FSM will do, since under FSM each decision involves simulating

a large number of hypothetical future trajectories. However, it may also be the case that a better learning procedure would be able to do significantly better job. That we get such good agreement in spite of this relatively low classification accuracy also suggests there are many situations that an agent can find itself in where the precise move it chooses is fairly irrelevant, at least in terms of the overall, qualitative swarm dynamics.

To test whether it was really necessary to use multiple hidden layers we also trained a network using exactly the same approach but with only one hidden layer made up of 250 neurons. In this case, we found that we were only able to achieve a classification accuracy of $\approx 38.5\%$, and whilst the resultant swarm produced was still qualitatively similar it was also significantly less stable — see SI movie 5 in the supplementary materials for a side-by-side comparison with the deep network.

At this stage we have a heuristic which mimics FSM, at least qualitatively, over a long period of time and which can be run with relatively little computational expense. Given the discussion in the previous section, a natural next step would be to run the full FSM algorithm but using this trained neural network to model the hypothetical future motion of the other agents. We could then use the results of these simulations to train a new neural network, and iterate this process to try and get some form of convergence towards self-consistency. In this case, we might hope to achieve convergence at the level of multiple quantities of interest (e.g. average order, opacity and density), rather than just the average order. Unfortunately we were not able to get this to work properly as we ran into issues training the second-level neural network (i.e. when we train it using data obtained from FSM with the first neural network used to generate the hypothetical trajectories of the other agents). Here, we found that we were not able to get such a high classification accuracy, dropping from $\approx 43.8\%$ to $\approx 38.0\%$, which significantly affected the robustness of the resultant swarms. This did not appear to improve with subsequent iterations, and so we were not able to demonstrate a clear convergence, nor have we really been able to explain the reason for this. It is possible that it is just fundamentally more difficult to predict the correct moves made by the FSM algorithm when it uses a more complicated heuristic for the motion of the other agents, but this is hard to verify without really knowing anything about how the neural network is learning. The assumption is that if we were able to train a neural network with a significantly higher classification accuracy, such that it is quantitatively more accurate in mimicking FSM, that we would be able to get this to converge.

4.5 The importance of collision detection

In the model as it has been introduced so far, as mentioned previously, when we consider the hypothetical future trajectories for an agent we cut off any branches where a collision is predicted to occur. Whilst this is quite natural, in this section we examine what happens if we remove this feature and instead just choose whichever move leads purely to the largest number of distinct future visual states. Intuitively, one might expect that this would not have a significant impact on the outcome, however surprisingly we find that this is not the case. When we run the algorithm with no collision detection we find that all of the agents come together to form an extremely dense swarm where all of the agents overlap significantly with each other (see figure 4.8(a)).

Intuitively it is clear that this state cannot be the one that maximises the average number of future visual states available to each agent. To demonstrate this, we consider running a simulation without collision detection but where we scale all of the resulting positions at each time step by a range of different factors. From each of these new configurations we then perform a future tree search (again ignoring collisions), counting the average number of distinct visual states available if the agents instead started from this scaled up configuration. Note that the actual trajectories are generated by FSM with no scaling — it is only afterwards that we visit each frame and recalculate what the number of future states available would be if we scaled the distances between agents. We see very clearly that the state which the swarm actually settles into is far from optimal in the sense of maximising its potential future state count, and that, if they were able to make decisions collectively, they could reach a configuration where they all have access to far more future visual states on average. However, because they are all moving by individually following a local gradient so as to “selfishly” maximise their own potential visual state count, they end up in a state which does not correspond to a global optimum. Figure 4.8(h) shows the result when future collision detection is accounted for and we see that they find themselves in a state which collectively has a much higher state count (although probably still not collectively optimal, as they still selfishly follow local gradients). This is true even if we count future visual states including collision detection, which can potentially cut off many future states. As such, it seems that somewhat fortuitously the inclusion of collision avoidance in the modelling of hypothetical future trajectories prevents the agents from selfishly moving into an extremely non-optimal global configuration, which subsequently leads to more interesting collective motion.

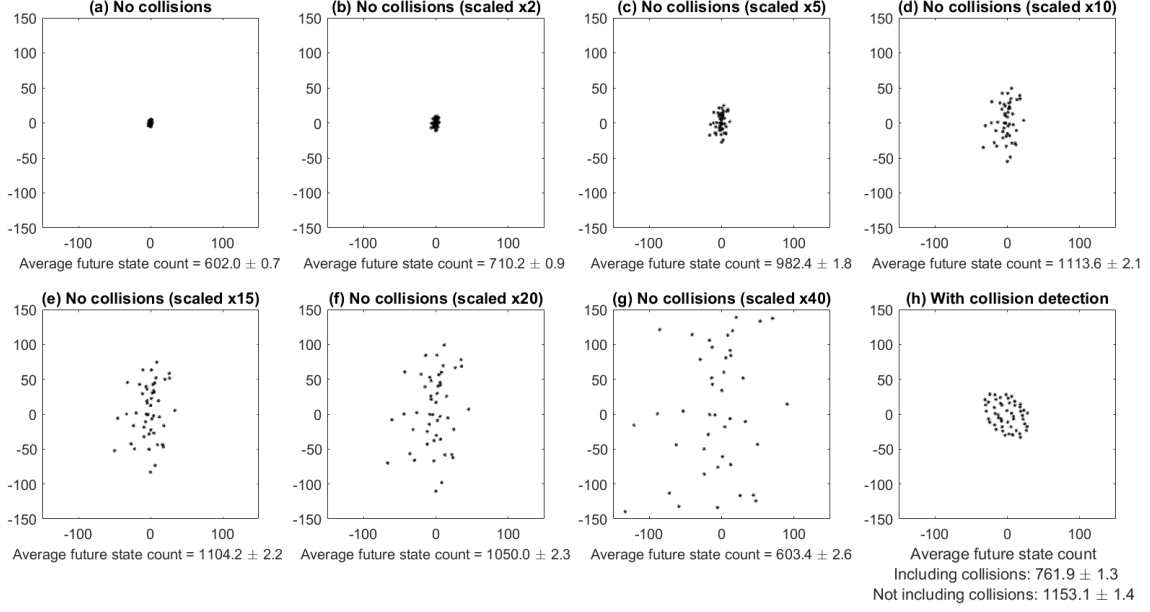


Figure 4.8: The importance of truncating hypothetical trajectories when a collision is detected. (a) A snapshot of the swarm which forms when we do not account for predicted future collisions. The agents come together and form a very dense swarm where they are constantly overlapping with each other. (b)-(g) show the same snapshot but where the distances between agents have been artificially scaled up by different factors, as indicated. For each time step of the simulation represented in (a) we carry out the same scaling and calculate the average number of future visual states available from these scaled configurations, rather than the actual configuration the system is in (using $\tau = 5$ and averaging over 2000 time steps). We see that the dense swarm which actually forms definitely does not maximise the agents' average future visual state count. (h) shows a swarm which is generated with the inclusion of collision detection in the hypothetical future trajectories. For each time step we then count the number of visual states available when we include (or don't include) collision detection. We see that even with the inclusion of collisions, and hence the truncation of branches of the future tree, there are more visual states available on average. A fairer comparison can be obtained by not including collisions when actually evaluating the future state count, in which case we see that there is a huge increase compared to (a).

4.6 Other variations in the model

4.6.1 Elliptical agents

So far we have only considered agents which are circular disks of unit radius. Whilst this has the advantage of extreme simplicity, it is of interest to see how the results of

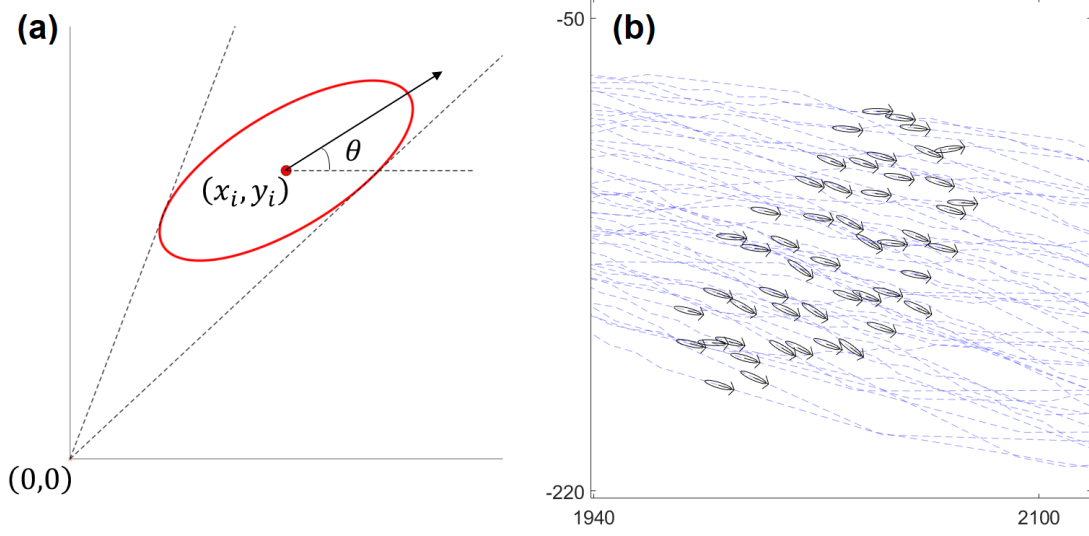


Figure 4.9: (a) Schematic for taking the visual projection of an elliptical agent placed at (x_i, y_i) (relative to the agent taking the projection), oriented at an angle θ . (b) A Snapshot of the trajectories followed by elliptical agents with a major axis of 3 and a minor axis of 1, moving according to FSM.

this model change when we consider agents with more complicated geometries. In this section we think about how we can generalise to elliptical agents, and analyse the effect this has on the resultant swarms. To do this the only real question we need to address is how to calculate the visual projection of an elliptical agent. Let us take the current position of the agent whose visual state we are considering to be the origin, and consider an elliptical agent placed at position $\mathbf{x} = (x_i, y_i)$, with its major radius a oriented along angle θ and its minor radius equal to b , as shown in figure 4.9(a). The equation of this ellipse is:

$$\frac{[(x - x_i) \cos(\theta) + (y - y_i) \sin(\theta)]^2}{a^2} + \frac{[(x - x_i) \sin(\theta) - (y - y_i) \cos(\theta)]^2}{b^2} = 1 \quad (4.2)$$

We then consider the interception of this ellipse with a straight line of gradient m passing through the origin, described by $y = mx$. Substituting this into the equation for the ellipse we can obtain a solution for the points where the line intersects the ellipse, in the form $x = \frac{x_{num} \pm \sqrt{x_{sqr t}}}{x_{den}}$. However, for the purposes of calculating the visual projection we are only interested in the cases when the straight line *just touches* the ellipse. In this case, there is only one point of intersection between the line and the ellipse, meaning that $x_{sqr t} = 0$. Running through this calculation we have a quadratic equation for the

two values of m that just touch the ellipse:

$$m = \frac{m_{num} \pm \sqrt{m_{sqr}}}{m_{den}}$$

where:

$$\begin{aligned} m_{num} &= (a - b)(a + b) \sin(2\theta) - 2x_i y_i \\ m_{den} &= a^2 + b^2 - 2x_i^2 + (a - b)(a + b) \cos(2\theta) \\ m_{sqr} &= 2b^2(x_i^2 + y_i^2) + 2a^2(x_i^2 + y_i^2 - 2b^2) \\ &\quad - 2(a - b)(a + b) [(x_i - y_i)(x_i + y_i) \cos(2\theta) + 2x_i y_i \sin(2\theta)] \end{aligned} \tag{4.3}$$

These values can then be used to find the actual points of intersection, and hence to calculate the angular interval which is filled by the agent.

If we fix the minor axis as equal to one and consider varying the major axis a of the ellipse (which is either in the direction of travel or perpendicular to it), then overall we find that up until around $a \approx 5$ the resultant swarm is qualitatively similar to when the agents are just circles — a snapshot of the trajectories with $a = 3$ is shown in figure 4.9(b). Increasing the value of a much beyond this leads to the swarm becoming unstable, at least for the values of τ we have considered. As such it seems like the qualitative results arising from FSM maximisation are partially effected by the geometry of the agents, although not too significantly.

4.6.2 Blind spot in the visual field of view

If order to make comparisons between our model and experimental data we may need to incorporate features which make it more realistic, or at least show that these features do not have an especially significant affect. In our model we have assumed that agents have full 360° vision (in 2D). This is certainly not true for any real animals that we may wish to compare with (and of course they also don't live in 2D), although we note that Starling's visual blind spot is actually fairly small at $\approx 64^\circ$ [57]. In this section we consider the effect of including a visual blind spot where the agents have a certain angle $\Delta\theta_{exc}$ behind them where they cannot see. This is shown schematically in figure 4.10(a). Since we are working with agents that have a discrete number of sensors there are two options we have here in terms of how to modify the visual states that the agents report. One is to simply leave the agent with the same number of sensors as it had before, but say that any of them which lie within the excluded region always read a zero. The other is that we scale up the number of sensors such that there are always

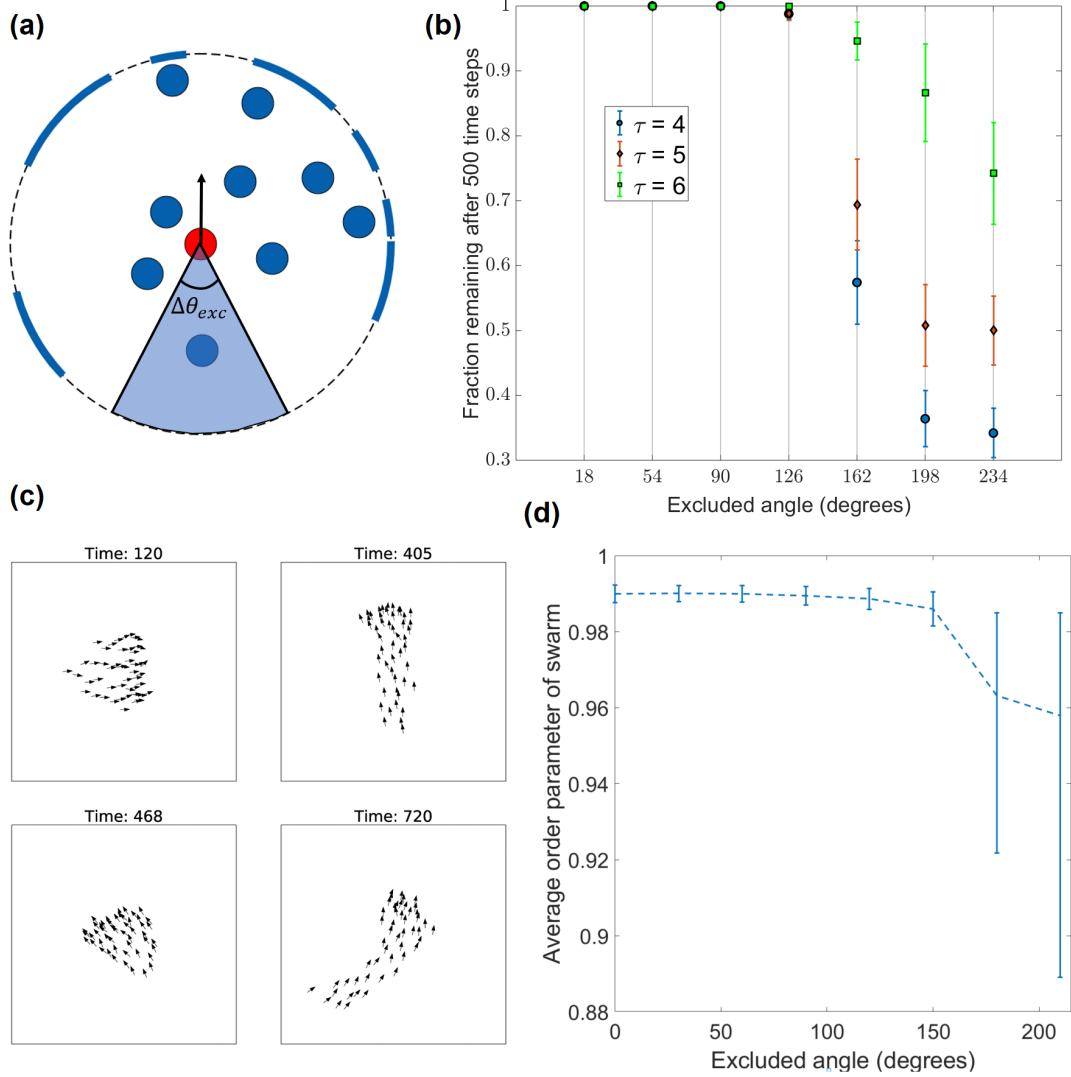


Figure 4.10: (a) Inclusion of an excluded angle $\Delta\theta_{exc}$, creating a region behind the agents where they cannot see. (b) Demonstration of the swarm becoming less stable as this excluded angle is increased. The fraction of agents left in the swarm, as defined in section 4.2, after 500 time steps is plotted for different values of $\Delta\theta_{exc}$ and τ . In this case the agents have the same number of sensors ($n_s = 40$), but any that fall in the excluded region always register a 0. Each point is averaged over 10 separate runs. (c) Snapshots of a swarm with $\Delta\theta_{exc} = 210^\circ$ at four different times. In this case the number of sensors is scaled up such that there are 40 sensors *in the non-excluded angular region*. This improves the stability. (d) The average order parameter of the main swarm vs. the excluded angle, using the second approach where the number of sensors are scaled up.

the same number, n_s , in the non-excluded region. For small values of $\Delta\theta_{exc}$ we find that this makes little difference, however as we look at larger excluded angles we find that the latter method is more stable. Figure 4.10(b) shows results from using the first of these approaches, looking at the fraction of agents which are still apart of the main swarm (as defined in section 4.2) after some number of time steps (500) when $\Delta\theta_{exc}$ is increased. We see that once we get to excluded angles in the region of $\sim 180^\circ$ that a significant fraction of the swarm fragments, although this can be mediated by increasing τ . In the second case where the number of sensors is scaled up, we find that for these excluded angles the swarm usually stays together. Nevertheless, if we look at the average order parameter (figure 4.10(d)) it is clear that the swarm is becoming noisier for larger values of $\Delta\theta_{exc}$. It is interesting that this seems to happen fairly abruptly, with the swarm being almost completely stable for excluded angles less than around 150° . Snapshots of a swarm using the second approach and with an excluded angle of 210° are shown in figure 4.10(c), where we see how it tends to fluctuate between being a swarm qualitatively similar to the model with full vision and being stretched out in the direction of travel. Intuitively, this makes some sense as it means that more of the agents are in the non-excluded angular region and so can actually be seen in the near future (agents are only looking $\tau = 5$ into the future and so can only reorient by a maximum of $\tau\Delta\theta = 75^\circ$ during the future trajectories being considered). We would hypothesize, and there is some evidence from the data, that as $\tau \rightarrow \infty$ the swarms will remain stable for any value of $\Delta\theta_{exc}$.

4.7 A more continuous measure of the degeneracy of visual states

The model that we've studied so far is inelegant insofar as we need to divide the agents' visual field of view into a discrete number of sensors, n_s . Whilst in reality all visual systems do involve a finite number of sensors (e.g. retinal cells are discrete), and although we have discussed some of the considerations that we need to make when choosing n_s , overall it is fairly arbitrary. As such, removing it as a parameter might make the model more elegant and would, at least, decrease the number of control parameters by one. In Section 3.2, we proposed a function that compares how different any two given states are. We then maximised the average pairwise difference between all of the accessible future states. If we continue to take the visual state to be defined in terms of a geometric projection of the other agents (as in figure 4.1(a)), but no longer divide it up into a finite

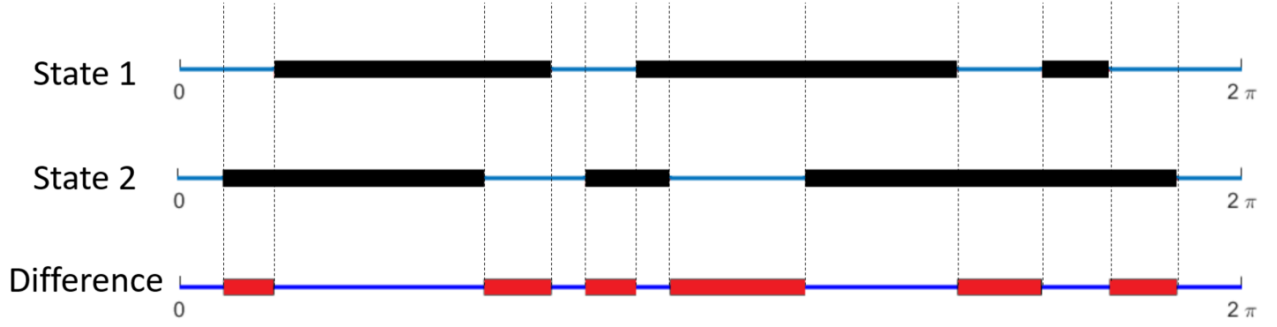


Figure 4.11: Defining a measure of the difference between two continuous visual states. A fully continuous visual state can be defined on the interval $[0, 2\pi)$ in terms of the edges obtained from the projection of the other agents. We can then simply define the difference between any two states to be the fraction of the interval where they are different.

number of sensors, we can define a completely continuous visual state on the interval $[0, 2\pi)$. By introducing a function $f(\theta)$ which takes on a value of 0 or 1 depending on whether the projection of any other agent occupies the agent's visual field at the angle θ or not, a natural measure of the difference between any two given visual states i and j can be defined in the following way:

$$d_{ij} = \frac{1}{2\pi} \int_0^{2\pi} [f_i(\theta) (1 - f_j(\theta)) + f_j(\theta) (1 - f_i(\theta))] d\theta \quad (4.4)$$

That is, the difference is defined as the fraction of the angular field of view where the two states take on *different* values. This is illustrated schematically in figure 4.11. We can then generalise the FSM maximisation algorithm by taking exactly the same approach, but now rather than counting the total number of distinct visual states available to each initial branch α of the future state tree we instead assign a weight to each node given by the *average difference between this state and every other state accessible on this branch*. That is, we build a tree of future states as before (with truncation on/after collisions), and for each of the α available initial moves that we are considering we will have a tree which has n_α possible states on it. We then define a score W_α for each of these initial branches in the following way:

$$W_\alpha = \frac{1}{n_\alpha} \sum_{i=1}^{n_\alpha} \sum_{j=1}^{n_\alpha} d_{ij} \quad (4.5)$$

The agents then choose whichever move corresponds to the branch with the largest score. Figure 4.12(a) shows snapshots of a typical trajectory of the resultant swarm generated by using FSM based on W_α . We see that this is qualitatively very similar to what we obtained in the discretised case (here $N = 50, \tau = 5$ and the other parameters are the “nominal” values — a video is included as SI Movie 6 in the supplementary materials).

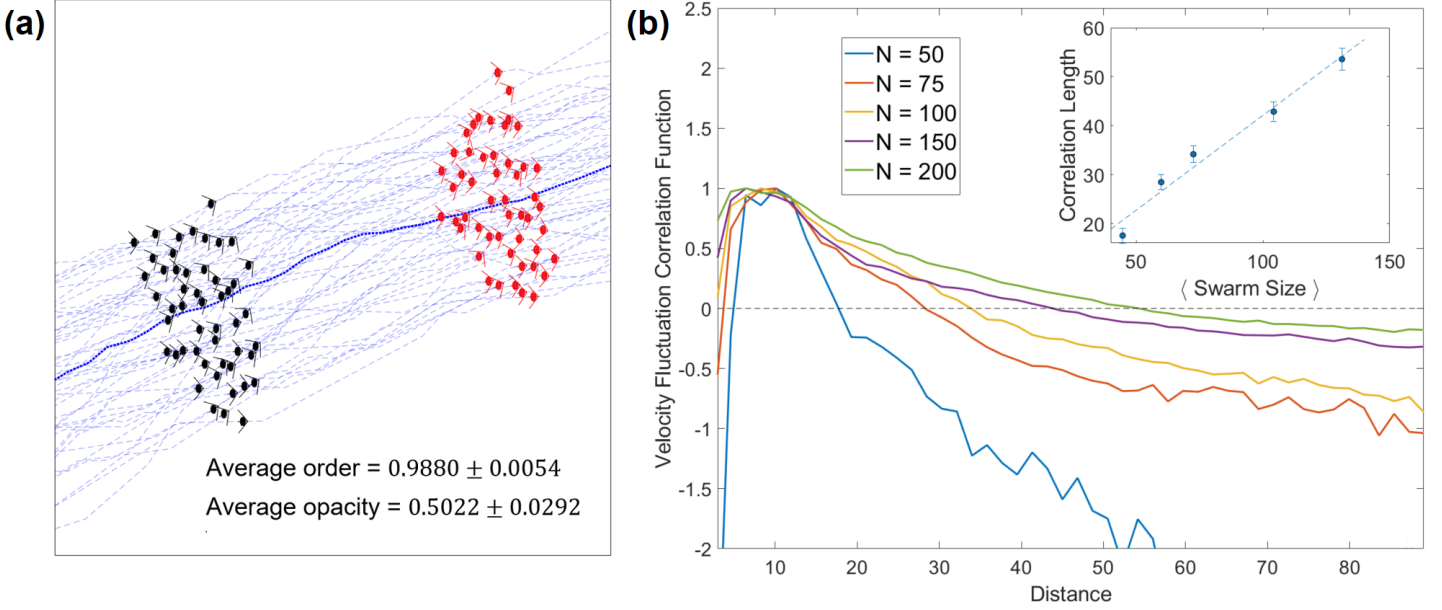


Figure 4.12: (a) Trajectories of a typical swarm that forms when we use the continuous measure of the degeneracy of future visual states, for $N = 50, \tau = 5$ and other parameters taking the standard values. See SI movie 6 in the supplementary materials. (b) Correlation functions for the fluctuations around the average velocity. These results are obtained from simulating swarms made up of different numbers of agents using the continuous measure of visual state degeneracy. This can be compared with figure 4.4(a). Here the normalisation is slightly different in that we divide each correlation function by its maximum value (not including self-correlations) — this simply makes it easier to display them all on the same plot. The inset figure shows how the correlation length (the point at which the function crosses the axis) varies with the physical size of the swarm. All results are averaged over 1000 time steps after the swarm has equilibrated.

We see that the average order parameter is again very high and that the average opacity of the swarm is extremely close to 0.5. This is more interesting than before, however, since we no longer have any discretised sensors with a given tolerance but instead have a fully continuous measure of the degeneracy of visual states. This provides strong ev-

idence that FSM naturally leads to swarms that are marginally opaque. We can also make a rough analytic argument that this should be the case. Starting from equation 4.5, the quantity that we are trying to maximise under the FSM algorithm is given by:

$$\langle d_{ij} \rangle = \frac{1}{2\pi n_\alpha} \sum_{k=1}^{n_\alpha} \sum_{m=1}^{n_\alpha} \int_0^{2\pi} [f_k(\theta)(1 - f_m(\theta)) + f_m(\theta)(1 - f_k(\theta))] d\theta \quad (4.6)$$

We can consider the functional derivative of this expression:

$$\frac{\delta \langle d_{ij} \rangle}{\delta f_p} = \frac{1}{n_\alpha} \sum_{k=1}^{n_\alpha} \sum_{m=1}^{n_\alpha} \int_0^{2\pi} [\delta_{kp}(1 - f_m) - f_k \delta_{mp} + \delta_{mp}(1 - f_k) - f_m \delta_{kp}] d\theta \quad (4.7)$$

If we set this equal to zero then we find that:

$$\frac{1}{\pi n_\alpha} \left[\sum_{k=1}^{n_\alpha} \int_0^{2\pi} (1 - 2f_k(\theta)) d\theta \right] = 0 \quad (4.8)$$

That is, on average $\int_0^{2\pi} (1 - 2f_k(\theta)) d\theta = 0$. This implies that:

$$\int_0^{2\pi} f_k(\theta) d\theta = \pi \quad (4.9)$$

which is exactly the condition we require for marginal opacity on average, over all of the future accessible states. Whilst this condition cannot possibly be sufficient to guarantee that $\langle d_{ij} \rangle$ will be maximised (we could have n_α identical half-full states, for example), it nevertheless indicates that it is a necessary condition. It should be noted that it is unclear how justified this procedure is, since $f(\theta)$ is not continuous, and so this should not be taken as a rigorous proof. We should also emphasise that this is a result about the *average over all of the accessible future visual states*, and as such does not necessarily imply that the actual swarm must be in a state of marginal opacity. Nevertheless, intuitively it seems reasonable to expect that the easiest way of achieving this condition averaged over potential future states is to start from a state which is itself close to marginal opacity. This is, indeed, what we find empirically.

Using this continuous version of the model we can also study the correlation functions and how the correlation length changes with swarm size, as we did previously in the discrete case. This is shown in figure 4.12(b) for swarms made up of between 50 and 200 agents. We can compare this with figure 4.4(a), showing the results for the standard

model with discrete sensors. We note that figure 4.12(b) is normalised differently: to display all the correlation functions on a single plot we have normalised them such that their maximum values are equal to one, however this is relatively unimportant since we are primarily interested in the correlation length, which is not effected by this scaling. We see that the feature of *scale-free correlations*, in which the correlation length scales with the size of the swarm, is still a feature of the model when we use this new measure of the degeneracy of potential future visual states. We also observe that the correlation functions are smoother than in figure 4.4(a), especially for larger swarms, and that there is no prominent bump at shorter distances which we had no clear explanation for in the simulation results obtained with discrete visual sensors.

Finally, the results found in section 4.5 about the importance of including collision avoidance in the model also carry over to this more continuous setting in more or less exactly the same way (the average score W_α is much lower when collision avoidance is taken out of the future trajectories, and the resultant swarm is very dense). Overall, despite the fact that this modification to the algorithm is actually quite significant the fundamental philosophy has not changed, and we find that the swarms produced are extremely similar to those generated using discrete sensor arrays, at least for small N . As we shall see, there are some significant differences to the swarm morphologies when N is made larger, which we explore in section 4.8.

4.7.1 More continuous action selection

We can also modify the model to make the choice of action somewhat more continuous. This can be done for either the continuous or the discrete sensor version of the model. To achieve this, rather than having a discrete set of moves that an agent can pick from at each time step we instead consider a maximum reorientation angle $\Delta\theta_{max}$ and a maximum difference in speed from v_0 , Δv_{max} . We then sample some number (N_a) of these initial moves at random, such that the reorientation angle is chosen uniformly from $[-\Delta\theta_{max}, \Delta\theta_{max}]$ and the speed difference from $[-\Delta v_{max}, \Delta v_{max}]$. For each of these initial moves, we then simulate N_t hypothetical future trajectories, where the agent's moves are chosen randomly at each step within the intervals given and the other agents move according to whichever heuristic we choose. Here, rather than an exhaustive "tree search" of possible trajectories we are instead performing a Monte Carlo simulation where we simulate a fixed number of random future trajectories. We then record the visual state at each step, and if a collision is predicted this trajectory is truncated. This allows

us to produce a score based on comparing the average difference (or absolute number) of these potentially accessible future visual states from each of the initial moves we consider, and choose whichever of these has the highest. We should note that this is not a fully continuous approach to choosing an action, which would allow for the agents to choose *any* action in the given range at each time step.

4.8 Simulation of larger swarms

Up to this point we have considered relatively small swarms, with a nominal value of $N = 50$. This is because the computational resources necessary to simulate the FSM procedure grow significantly with both the swarm size, as well as with τ . To be more precise, there are N agents which make a decision at each time step by simulating a number of hypothetical trajectories, calculating a visual state at each possible future node. There are potentially $\sum_{i=1}^{\tau} N_m^i$ future nodes to consider, which is dominated by N_m^{τ} , where N_m is the number of moves each agent has available to it. The calculation of a single visual state involves calculating the $2N$ edges which then need to be inserted into the final visual state, which can be done in $O(N \log(N))$. Thus, the whole process is $O(N^2 \log(N) N_m^{\tau})$ per time step. Empirically, this means that simulating more than about $N = 200$ agents with $\tau = 5$ was not realistic when running the code on a single core. Despite this, even at these small sizes we find some interesting features in terms of the qualitative behaviour of the swarms as N is increased. Figure 4.13 shows how the shape of the swarm changes as N is increased (here using $\tau = 5$, as for $\tau = 4$ and $N = 200$ the swarm can be unstable and fragment). We see that as N is increased the swarm stretches out in the direction perpendicular to the centre of mass velocity.

To look at larger swarms than this it was necessary to parallelise the code so that it could be ran on many cores at once. The most natural way to do this was to use the MPI (Message Passing Interface) framework[58], assigning a certain number of agents to each core. That is, we have a primary process which, at each time step, broadcasts the current state of the swarm to the other cores, which then carry out the FSM calculation for the set of agents to which it has been assigned. Each of these cores then sends the decisions it has calculated back to the primary process, which updates the global state of the swarm for the next time step. Making use of Warwick University’s high performance computing facilities we were able to run a simulation of $N = 500$ agents with $\tau = 6$, making use of 251 cores — so 2 agents per core, plus one to receive the results and update/broadcast the state of the swarm. Snapshots of the results obtained from this

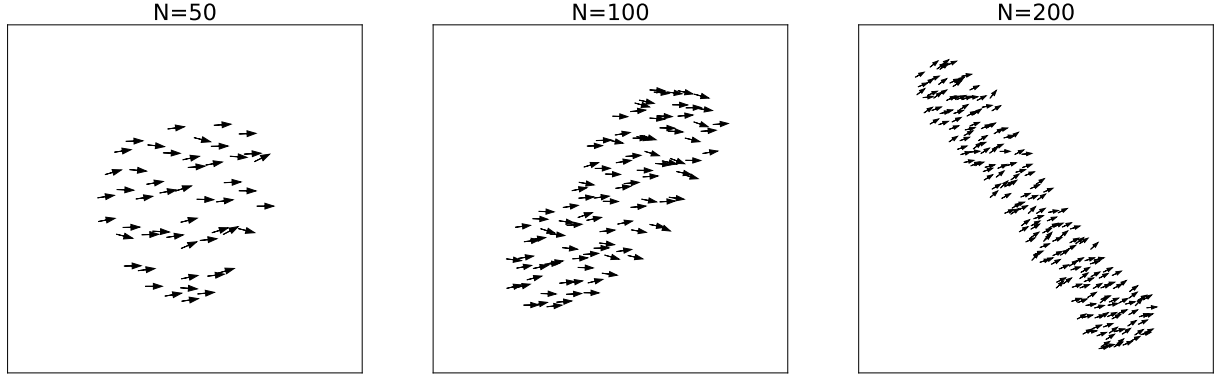


Figure 4.13: Snapshots of swarms made up of different numbers of agents, using discrete visual states. Here we use the standard parameters (apart from τ being changed to 5 to increase stability). As N is increased, the swarm tends to stretch out in the direction perpendicular to the direction of movement.

are shown in figure 4.14. The system starts with the agents positioned randomly in a square region before going through a number of transitory states involving rich dynamics. Eventually the swarm then fragments, splitting into two subgroups which are both stretched out perpendicularly to the direction of travel. A video of this is included in the supplementary materials as SI movie 7.

To explain why the swarm tends to stretch out in this way we thought about how the tree of accessible final states looks for different choices of parameters, shown in figure 4.15. Here we consider varying $\Delta\theta$ and Δv , which we see has a significant impact on the set of final positions that can be reached with the allowed moves after τ time steps. Here we have used $\tau = 5$ and otherwise standard parameters. Note that obviously v_0 also effects the final positions that can be reached, however we have kept this as a constant. We see that for the “standard” set of parameters where $\Delta\theta = 15^\circ$ and $\Delta v = 2$ the distribution of final positions is stretched out in the direction perpendicular to the agent’s current orientation. We then see that increasing $\Delta\theta$ increases this effect, as does decreasing Δv . We might therefore expect that this is the explanation for why the swarm stretches out in a similar way, and that when we decrease $\Delta\theta$ or increase Δv that the shape of the resultant swarm should change as well. The results shown in table 4.1 show that in fact this is not the case. Here we report the average size of the flock in the direction in which it is largest (as measured by the intersection of an infinite line

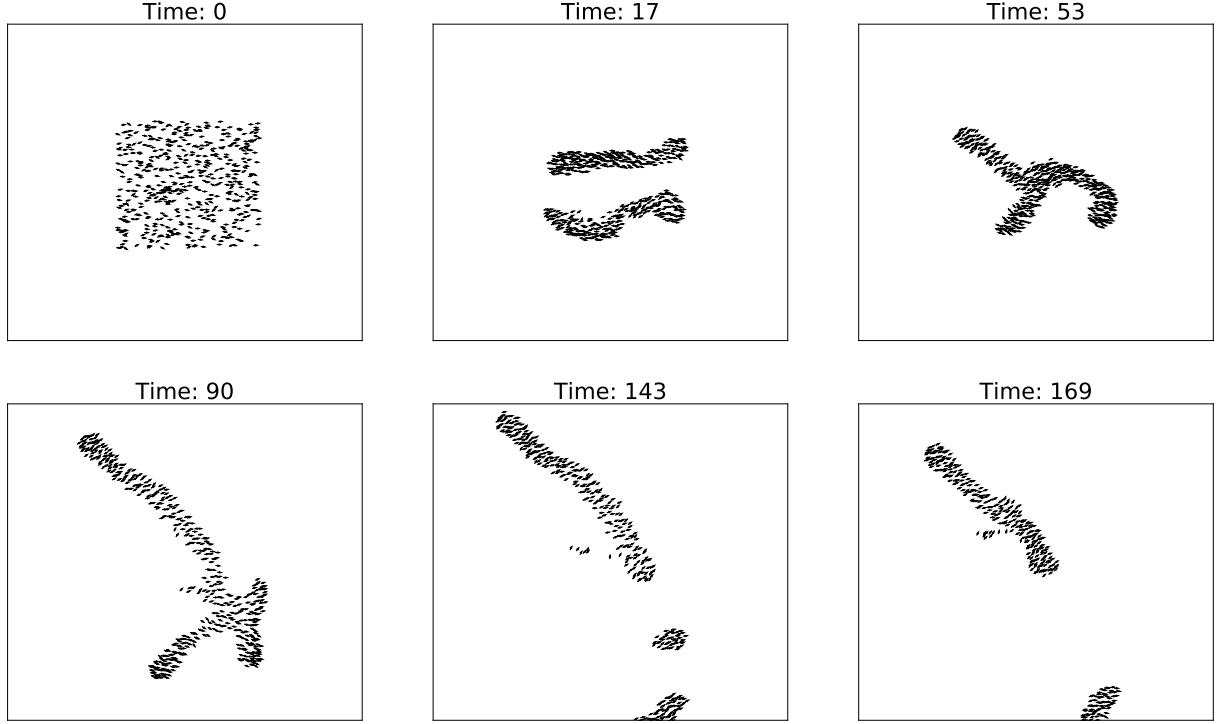


Figure 4.14: Snapshots of a simulation made up of $N = 500$ agents (with $\tau = 6$) at six different times, shown in a frame co-moving with the swarm's centre of mass. The agents start in an initial region of size 400×400 where they are placed at random, and evolves according to the standard FSM algorithm with discrete visual states.

with a given orientation and the swarm's convex hull), as well as the direction in which it is smallest, and the ratio between the two (essentially an aspect ratio). We see that whilst increasing Δv does lower this ratio slightly, lowering $\Delta \theta$ actually increases it, and in all cases the resultant swarms are still stretched out significantly. A clear, intuitive reason for why this should be the case is still illusive.

We can compare these results with what happens if we instead use the continuous measure of visual state degeneracy discussed in the previous section, as opposed to the

	Maximum Distance	Minimum Distance	Ratio
$\Delta\theta = 15^\circ, \Delta v = 2$	294 ± 29	39.5 ± 4.5	7.5 ± 1.2
$\Delta\theta = 10^\circ, \Delta v = 2$	276 ± 31	30.4 ± 4.3	9.3 ± 1.7
$\Delta\theta = 5^\circ, \Delta v = 2$	258 ± 68	18.5 ± 2.7	14.3 ± 4.0
$\Delta\theta = 15^\circ, \Delta v = 1$	328 ± 44	31.8 ± 5.7	10.7 ± 2.5
$\Delta\theta = 15^\circ, \Delta v = 4$	305 ± 15	45.9 ± 5.3	6.7 ± 0.8
$\Delta\theta = 15^\circ, \Delta v = 6$	316 ± 28	54.0 ± 6.1	5.9 ± 0.6

Table 4.1: Results from running a swarm of $N = 200$, $\tau = 5$ and averaging over 1000 time steps. The maximum distance is taken to be the largest distance between any two agents in the swarm, and the minimum distance is calculated as the minimum difference between the two points of intersection between the swarm’s boundary and a line passing through its centre of mass (minimised over all possible orientations of this line).

standard model with discrete sensors. In figure 4.16 we show snapshots of the swarm produced for $N = 50, 100$ and 200 . Interestingly we see that the morphology of the $N = 200$ swarm is significantly different from the discrete sensor case in that it is more compact and not stretched out as much in the direction perpendicular to the swarm’s motion.

The code for simulating this model can also be parallelised using exactly the same approach as for the discrete sensor case. Snapshots of a simulation with $N = 500$ and $\tau = 5$ (not $\tau = 6$ as in the discrete case, as this was too slow to run) are shown in figure 4.17 at six different times. Again the agents are initially placed at random within a square region of size 400×400 and evolve according to the FSM algorithm. We see that the swarm’s morphology changes over time, and the dynamics that result are extremely rich, with the swarm tending to oscillate between being somewhat stretched to being more compact. A video is included in the supplementary materials as SI movie 8. It is quite remarkable that such complex dynamics have emerged from a model where the decisions are made based on a single underlying principle, especially as we have confined ourselves to motion in a two-dimensional plane. We might expect that if this model were to be extended to 3D that the resulting behaviour would be even more complex.

4.9 Controlling the swarm

We have already noted the important role played by the heuristic used to model the motion of other agents in the hypothetical future trajectories. In this section, we look at how this can be adjusted in a way that allows us to guide the swarm to follow any

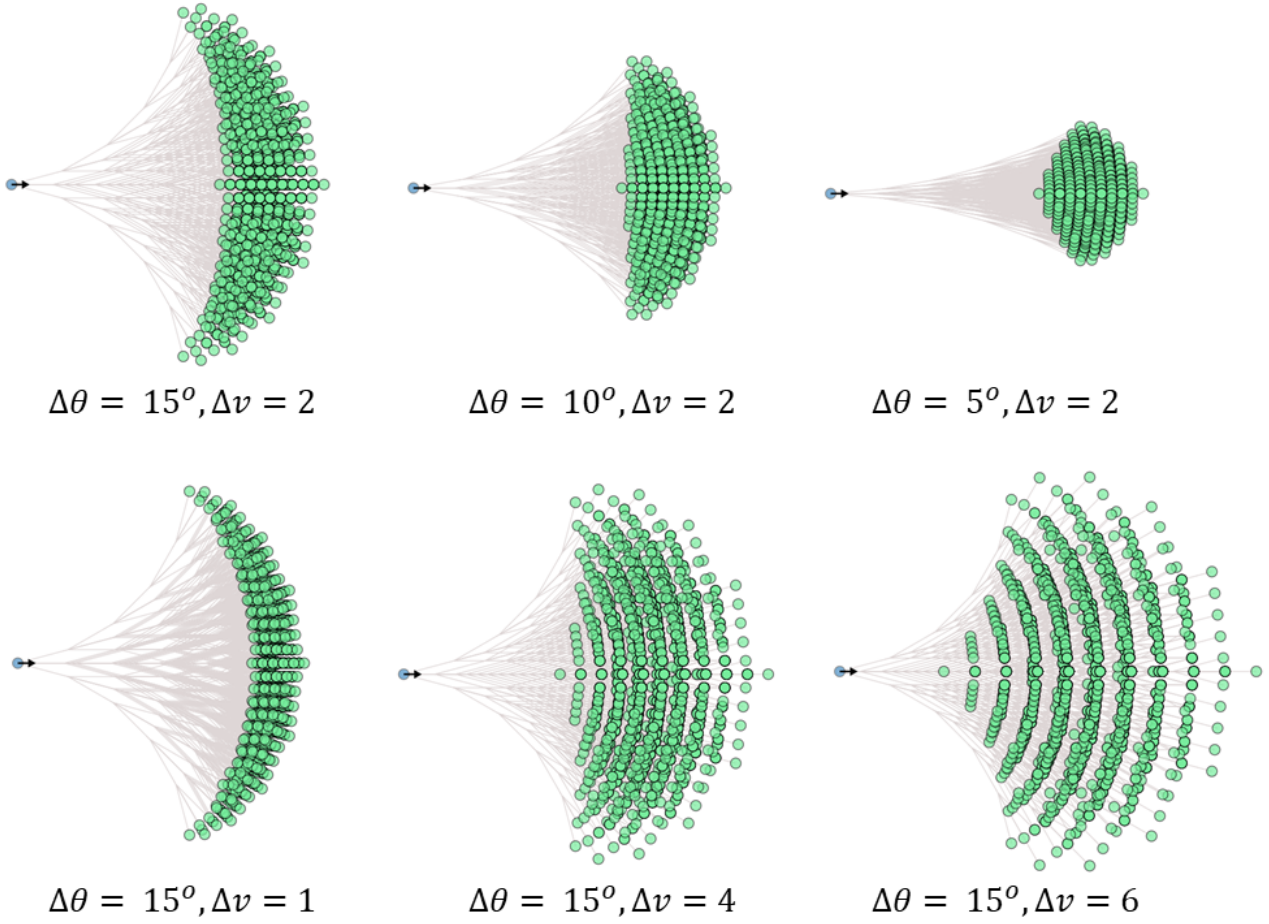


Figure 4.15: Distributions of accessible final states which can be reached for different parameter values. Here we show the positions of the final states which are accessible to an agent which analyses out to $\tau = 5$ time steps into the future. This distribution is determined by the moves the agents have available, and so depends on $\Delta\theta$ (how much an agent can reorient itself in one time step), Δv (how much an agent can change its speed) and v_0 (the agent's nominal speed). We might expect that changing these parameters could have a significant impact on the morphology of the swarm, however the results in table 4.1 suggest that actually this effect is surprisingly small.

arbitrary trajectory, provided that its curvature is not too high (how quickly the swarm can turn depends on the reorientation angle $\Delta\theta$). All of the work in this section was carried out using the discrete visual-sensors version of the model however the results should carry over to the continuous version as well.

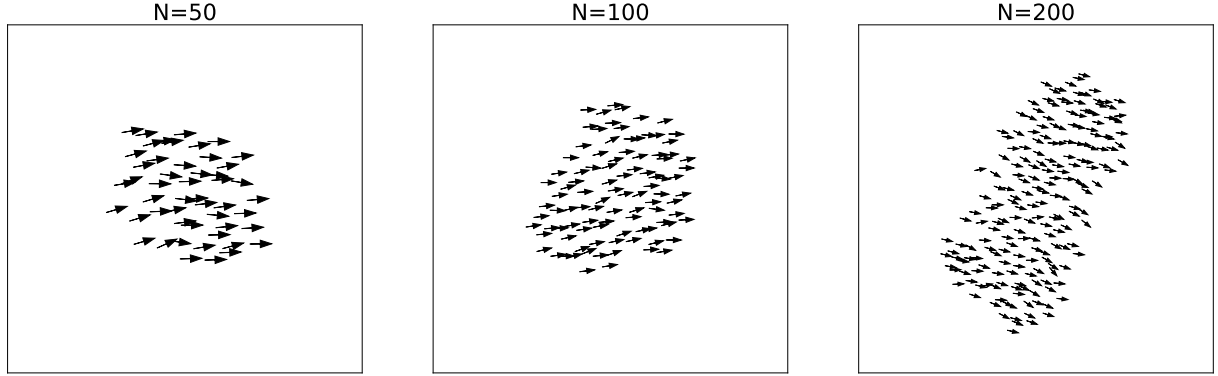


Figure 4.16: Snapshots of swarms made up of different numbers of agents when we use the continuous measure of the degeneracy in potential future visual states. The parameters here are standard except for $\tau = 5$. The morphologies of the swarms can be compared with the discrete sensor case shown in Figure 4.13.

4.9.1 Controlling the swarm trajectory

If we want to control the swarm there are two possible approaches we could take. The most naive approach would be to carry out the FSM procedure using the ballistic heuristic for other agents and then simply add an amount $\Delta\theta_t$ to the orientation of each agent after they’ve made their decision. Here $\Delta\theta_t$ is a turning angle which we can relate to a turning radius R_t (assuming agents move at speed v_0) as follows:

$$R_t = \frac{v_0}{\sin(\Delta\theta_t/2)} \quad (4.10)$$

The alternative approach is to adjust the heuristic which is being used to model the motion of agents in the future by encoding an expectation of turning. That is, when making a decision rather than assuming that all of the other agents will move in a straight line we instead assume that they will all be changing their orientations by an amount $\Delta\theta_t$ at every future time step. We then proceed to choose the move which leads to the largest number of future states under this assumed heuristic. Intuitively, we know that maximising one’s access to future visual states should require remaining in the vicinity of where one expects the other agents to be, and so this should naturally lead to the swarm turning, becoming in effect a “self-fulfilling prophecy”. If the resultant swarm does indeed turn, we also have the obvious advantage that agents are then using a more self-consistent heuristic for the future motion of the other agents.

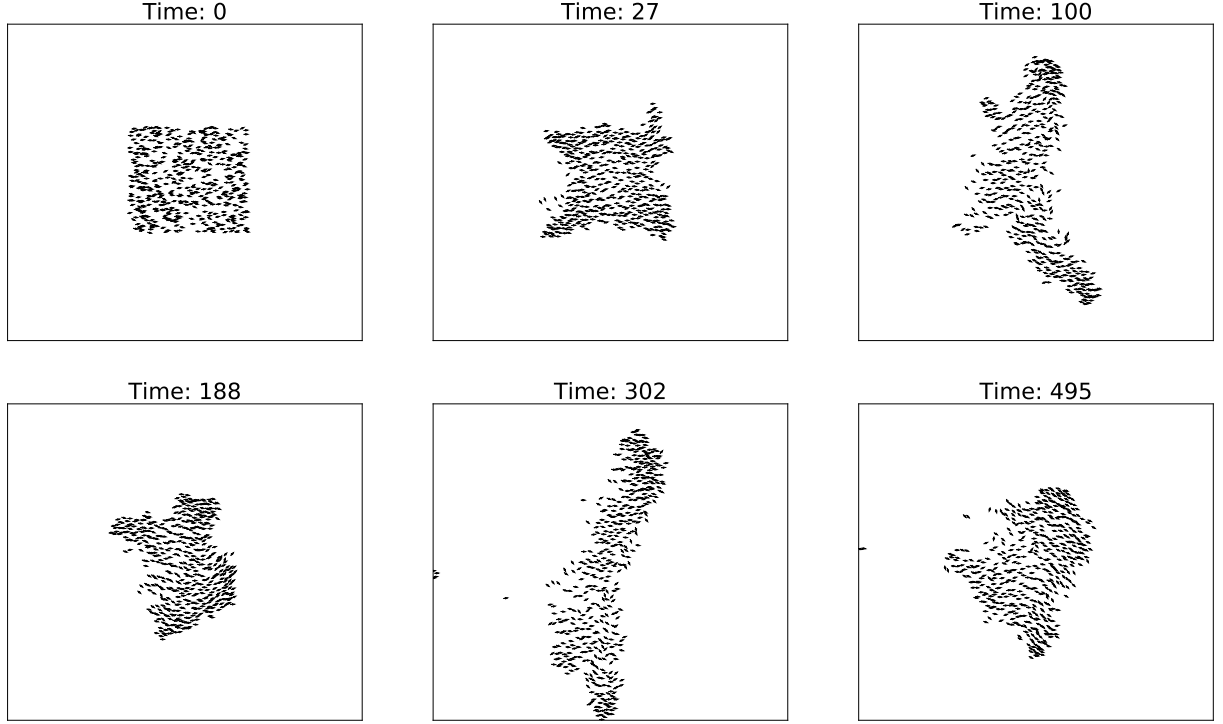


Figure 4.17: Snapshots of a simulation made up of $N = 500$ agents (with $\tau = 5$) at different times, shown in a frame co-moving with the swarm’s centre of mass. The system starts from an initial square region of size 400×400 where agents are placed at random, and evolves according to the FSM algorithm using the continuous measure of visual state degeneracy described in section 4.7. This can be contrasted with the results of using discrete sensors shown in figure 4.14.

Figure 4.18 shows a comparison of the two approaches for a turning angle of $\Delta\theta_t = \Delta\theta/4$, as well as a red circle of radius R_t for comparison. We find that in both cases the swarm does indeed turn, however only the second approach leads to it turning self-consistently with the actual rate that’s encoded: in the naive case (figure 4.18(b)) the turning rate is significantly lower. The explanation for this is that in the “naive” case the agents are positioning themselves having made a plan based on exploring τ steps out into the future, whilst assuming that the other agents will move ballistically. When they all have

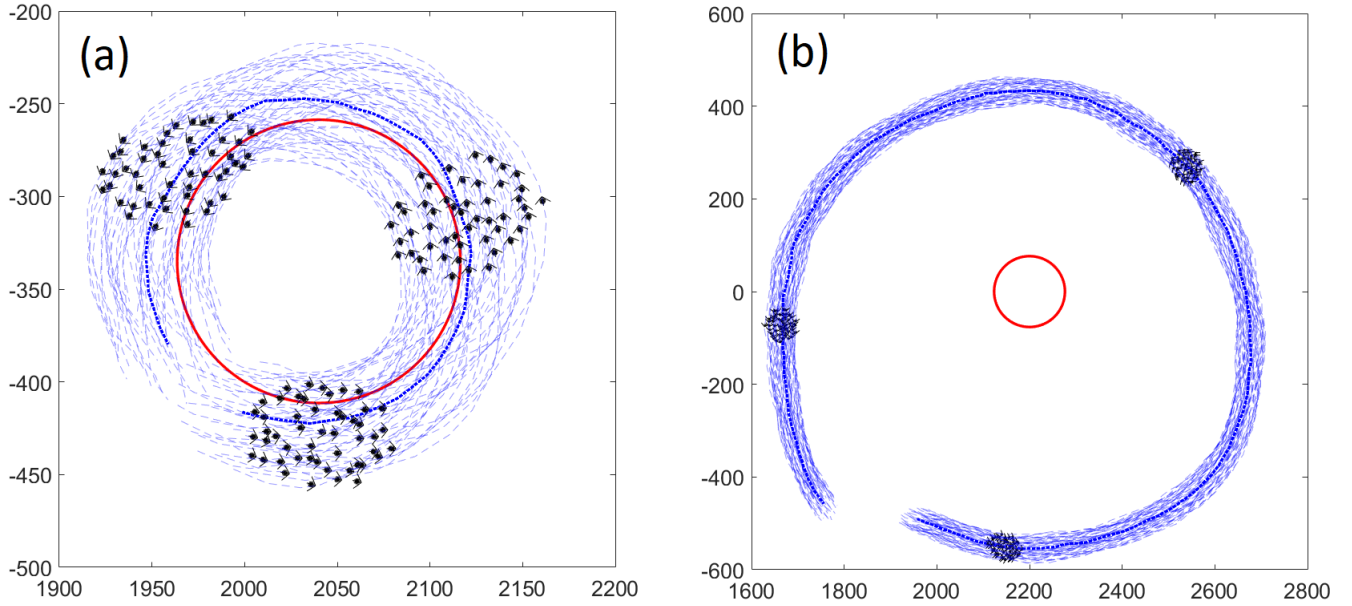


Figure 4.18: (a) Trajectories of a turning swarm when the agents update the heuristic they are using such that they expect the other agents to turn in the future. (b) The more naive approach in which agents make their decision based on FSM using the ballistic heuristic, but then add $\Delta\theta_t$ to each of their orientations after they've each made their decisions. $\tau = 5$ in both cases. The solid red circle shows the expected turning radius for the given value of $\Delta\theta_t$. A video comparing the two approaches side-by-side included in the supplementary materials as SI movie 9.

their orientations changed this is not consistent with what they had expected and so they partially resist this turn, making it slower. We believe that it is therefore preferable to use the approach in which the adjust the agents' heuristics, and so from now on that is the approach that we consider.

Using this approach we can, for a given set of parameters, determine an empirical maximum turning angle at which the resultant flock remains stable. We choose the "standard" set of parameters that we have been using throughout, but change τ from 4 to 5, as this leads to better stability for larger turning angles. Clearly the maximum stable turning angle cannot exceed $\Delta\theta_t$, and whilst we haven't conducted rigorous tests we find actually that making it significantly larger than about $\Delta\theta/4$ generally leads to the swarm becoming slightly unstable.

4.9.2 Guiding the swarm to follow an arbitrary trajectory

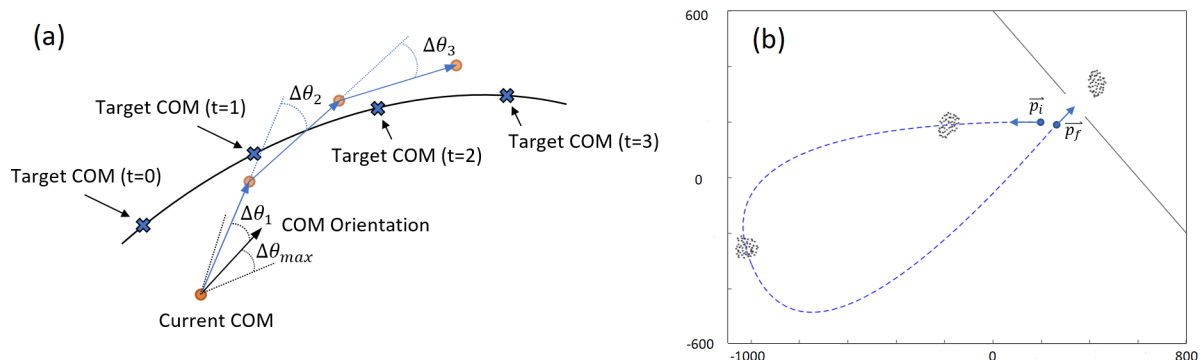


Figure 4.19: (a) Guiding a swarm to follow a trajectory of our choice. Having identified points on the target trajectory that we would like the swarm's centre of mass to occupy at time steps $t = 0, \dots, \tau$, we choose angles $\Delta\theta_1, \dots, \Delta\theta_\tau$ which are used in the heuristic to model hypothetical future trajectories to turn all of the other agents. These are chosen such as to move the swarm's centre of mass as close to the corresponding target position as is possible, within the constraint that the turning angle must be less than $\Delta\theta_{max}$. (b) An example of a trajectory generated from an initial point \mathbf{p}_i to a final point \mathbf{p}_f , with given final initial and final orientations. This is generated such that nowhere does its curvature exceed the value corresponding to $\Delta\theta_{max}$, meaning that the swarm is able to follow it without becoming unstable.

Now that we are able to make the swarm turn we are also in a position where we can guide it to follow (approximately) any trajectory that we choose. The ability to control the macroscopic properties of a swarm (in this case the centre of mass trajectory) could potentially be of interest in the design of artificial swarming systems, e.g. in the field of swarm robotics[40]. To do this we take the empirical maximum turning angle we have found, $\Delta\theta_{max}$, and use this to steer the swarm towards whichever trajectory we specify. That is, given the swarm's current configuration, we consider a heuristic for the other agents where at each future time step i they all turn by the same angle $\Delta\theta_i \in [-\Delta\theta_{max}, \Delta\theta_{max}]$. This angle is chosen such that the swarm's centre of mass moves as much as possible towards the desired future position on the trajectory that we wish to follow. The basic idea is shown schematically in figure 4.19(a). Taking the swarm's current centre of mass and centre of mass orientation, we determine an angle $\Delta\theta_1$ within the interval $[-\Delta\theta_{max}, \Delta\theta_{max}]$ that would move the centre of mass closest to the target at $t = 1$. This then determines the angle by which every agent will turn in the heuristic used to model future trajectories during the first future time step. This

will move the hypothetical swarm to a new centre of mass, where a new angle $\Delta\theta_2$ can be determined to move the centre of mass as close to the target at $t = 2$ as possible. This is repeated out until τ future time steps have been modelled. The simplest way of choosing the target positions for the centre of mass is to choose the closest point on the planned trajectory as the target for $t = 0$ and then the next target point for $t = 1$ at a distance v_0 further along the target trajectory. If the trajectory crosses itself, then some other criteria involving the swarm’s centre of mass orientation may also be needed to automatically calculate the target positions along the trajectory.

Following this procedure should allow the swarm to follow any trajectory where the maximum curvature does not exceed $\frac{\sin(\Delta\theta_{max}/2)}{v_0}$. If we have an initial point and an initial orientation, and a final point and final orientation, we can make use of an algorithm[59] which constructs two continuously joined quadratic Bezier curves between the two points with the required orientations and which never exceeds a specified curvature Γ . Whilst the curve produced with this algorithm is not optimally short, it is constructed in a way that penalises being excessively long. An example of such a curve generated in this way to guide a swarm through a gap is shown in figure 4.19(b). A video is also included in the supplementary materials as SI movie 10.

Being able to control a swarm in this way can be thought of as an example of “guided self-organization”[23], where we are able to influence the complicated “microscopic” interactions between agents in such a way so as to achieve an emergent collective behaviour with desired macroscopic properties that we specify.

4.9.3 Future state maximisation at different scales

In the example in the previous section we chose a trajectory for the swarm’s centre of mass to follow for a duration of time into the future which far exceeded τ , the number of time steps individuals use to model the trajectories necessary to make their decisions. This motivates us to think about a situation where we could apply the principle of future state maximisation to the generation of the centre of mass trajectory itself, and which could take place at a much larger time/length scale than that of the individual agents’ future state maximisation.

To do this, we consider exploring a coarse-grained tree of possible future centre of mass trajectories. To apply FSM at the level of the swarm’s centre of mass trajectory, rather

than counting distinct visual states we can instead count the number of different end points that are available on this tree, for each of the initial branches (or “meta-actions”). This means that here we are considering a measure of spatial freedom rather than a freedom associated with the number of possible visual states. To construct a tree which can explore much larger length scales, we define “meta-actions” which last many times longer than the individual time steps over which agents make their decisions. For simplicity, we allow for only three different meta-actions at each of the τ_m future meta-timesteps (where these consist of $n_{step} = 10$ individual time steps), and correspond to moving straight or turning with a constant curvature $\pm \frac{\sin(\Delta\theta_{max}/2)}{v_0}$ whilst travelling at speed v_0 . This is visualised in figure 4.20(a). We generate a full trajectory by choosing an initial move that maximises the number of end points on the tree, adding that to the current trajectory, and then repeating the calculation as many times as we choose.

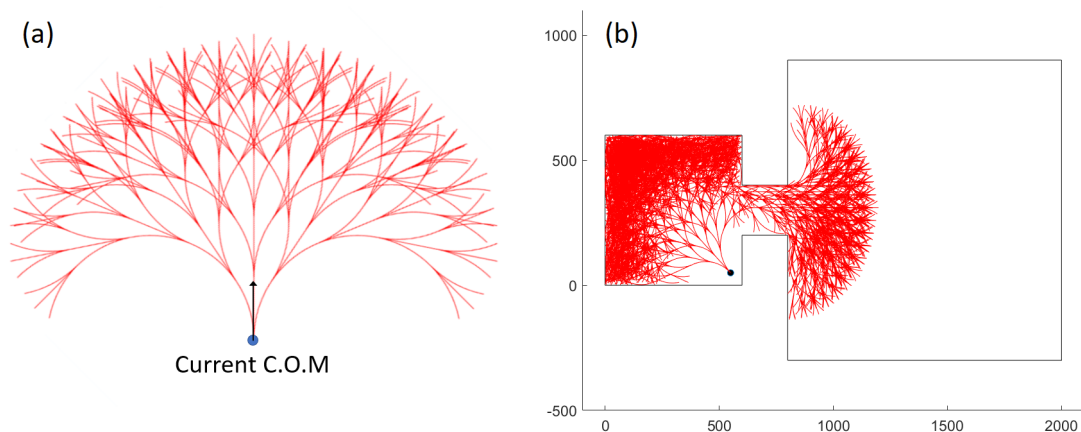


Figure 4.20: (a) A coarse-grained tree search over three “meta-actions”: moving straight or turning constantly with the swarm’s maximum curvature, to generate a centre of mass trajectory for a swarm. Here we consider sequences of $\tau_m = 5$ meta-actions, each of which lasts as long as 10 individual time steps. (b) An example of this tree in a geometry consisting of two different-sized, connected boxes. Branches of the future tree which collide with the walls of the geometry are cut off, and a gradient which drives the system towards regions where there are more available end points is followed, resulting in the swarm moving to the region where it experiences minimal confinement (maximal freedom). Here $\tau_m = 10$.

As a concrete example we consider placing a particle representing the swarm’s centre of mass within two connected boxes, as shown in figure 4.20(b). The figure also shows a visualisation of the initial tree of future states when we consider looking 10

meta-timesteps into the future, each of which corresponds to 10 individual time steps (and so cover $10v_0 = 100$ units of distance each). Branches of this future tree which collide with the walls of the geometry are truncated, and so if we look far enough into the future there are far more available trajectory endpoints within the larger of the two boxes. This naturally leads to a gradient which drives the particle into the larger box. Generally, the trajectory will then spend most of its time in the larger box, although it can occasionally venture back into the smaller box for short periods of time (this happens less frequently when more meta-timesteps into the future are considered). A video that shows a swarm which follows the trajectory generated in this way is included in the supplementary materials (SI movie 11).

As a larger scale example we then consider placing a swarm within a maze-like geometry, shown in figure 4.21. Here we initially consider looking 20 meta-timesteps into the future, where again each of these is equivalent to 10 individual time steps. Figure 4.21(b) shows the trajectory the centre of mass follows when we generate it in this way, as well as some snapshots of a swarm which is guided to follow it (videos of the evolution of the centre of mass tree search as well as the swarm following this trajectory are included in the supplementary materials as SI movies 12 and 13 respectively). Note that the geometry is padded out in figure 4.21(b) because the trajectory generated is for the swarm's centre of mass, which doesn't account for the finite size of the flock — adding a buffer prevents any collisions between individual agents and the walls. We see that even though the initial tree search is not quite long enough to explore the area in the top-right, which intuitively has the largest freedom, the generated trajectory still manages to find its way there. This is because the future time horizon that we use in the modelling of the available future trajectories is large enough so that all of the possible dead ends the agent could find itself in are detected, and hence avoided. Once the trajectory reaches the top-right hand corner it simply stays there and circles around, remaining close to the middle of the box.

Whilst there is no clear way to quantitatively link the maximisation of the number of end points for the centre of mass to the maximisation of the number of possible future visual states which motivates the individual agents, this serves as a nice example of how the principle of FSM can be applied to a system at two vastly different scales. Here the individuals are making decisions based only on what they predict for the next 5 time steps, but collectively their motion is based on a modelling a time horizon of 2000 time steps.

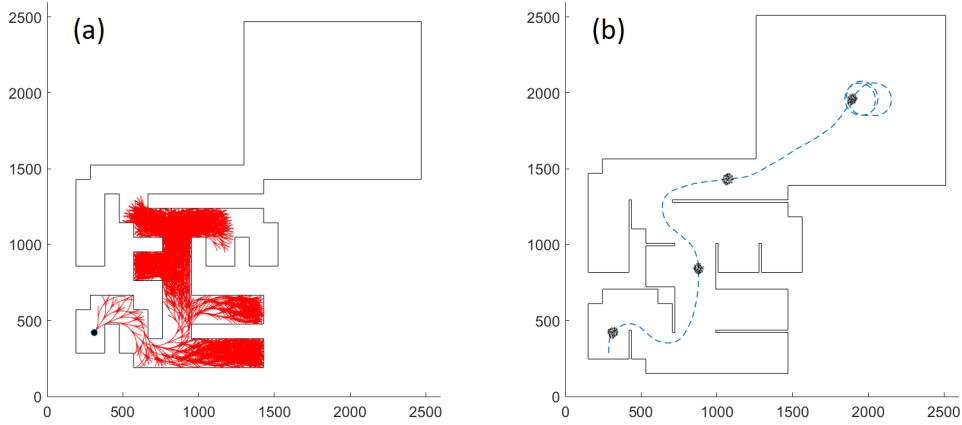


Figure 4.21: Guiding a swarm to “solve” a maze. (a) A visualisation of the future tree search used (initially with $\tau_m = 20$). Following a gradient in the number of available end-points generates the centre of mass trajectory shown in (b). This is able to reach the large area in the top-right since all of the dead ends are detected by the modelled future trajectories — and so the agent chooses a path which avoids these. (b) The actual centre of mass trajectory along with some snapshots of a flock which is using it as a guiding trajectory.

Overall, the primary contribution of this chapter has been to introduce a model of collective motion based entirely on the principle of FSM. The approach we have taken differs significantly from many of the well known models in the literature in that we start purely from a low-level motivational principle (FSM), allowing everything else to emerge spontaneously. We believe that this increases our model’s explanatory power. That we are able to get so much out of a model based on a simple principle (a robust, cohesive, highly aligned swarm, marginal opacity and scale free correlations) is interesting to us, especially given how general the principle of FSM could potentially be. As such, it is our hope that this work can serve as an illustrative example of the way in which FSM can be applied to explain and understand the behaviour of a potentially wide range of biological systems.

To close this chapter, we briefly comment on the feasibility of extending this agent-based FSM approach to a continuum model. There exist a number of analytical approaches that allow us to write down hydrodynamical equations for many-body dynamical systems, such as Boltzmann kinetic theory and the BBGKY hierarchy[60], as well as ap-

proaches based purely on symmetry (e.g. the Toner-Tu continuum equations that give a hydrodynamical description of the Vicsek model[61]). The difficulty with applying these to our model is that the interactions are extremely non-local, both temporally (in that each agent has to model many different future trajectories) and spatially (with visual interactions occurring over infinite distance). This makes writing down any kind of continuum model highly non-trivial using the standard approaches — however it would be an interesting area of research in the future to see if a local decision rule that can approximate this kind of interaction could be written down, which then might allow for a set of hydrodynamical equations to be derived. In a sense the neural network trained to mimic the FSM algorithm did something along these lines (at least in terms of reducing the decision rule to being more local in time — i.e. not requiring the modelling of future trajectories), however the resultant decision rule was not analytic and not easily interpretable, and regardless, still operated on the non-local visual field.

Chapter 5

Threat avoidance by active particles

Previously the systems we have considered have all been discrete in some way. A majority of the time we have been dealing with discrete state and action spaces, and whilst we have considered some ways of applying the principle of FSM when these spaces are made continuous, time has remained discrete throughout. That is, we have considered decisions to be made at the end of discrete intervals, and have modelled future trajectories as being composed of a finite number of these intervals. In this chapter we study a simple example of a decision making process in which both time, as well as the state/action spaces are taken to be continuous. We consider a single Brownian particle moving in one dimension, capable of using energy so as to generate an advective drift velocity. At the origin we place a stationary threat, which kills the particle upon contact. The question we ask is, “how should this drift velocity be chosen so as to balance the energy cost associated with moving and agent’s survival”. One way to do this is to apply the principle of FSM - i.e. to say that the particle should aim to maximise the expected number of future states/paths which are available to it. This provides the motivation for the approach we take to this problem, however what we actually do has significant differences with the methods used in previous chapters. Rather than directly trying to enumerate the possible future paths, we instead derive a “fitness functional” which we argue approximately represents the expected number of future paths available over the rest of the particle’s lifetime. The agent then chooses a drift-velocity $v(t)$ which maximises this quantity over the modelled hypothetical future trajectories. This determines the decision made at the present time (i.e. $v(0)$ gives the drift velocity now). This, in principle, can then be repeated at arbitrarily small intervals in order for the agent to

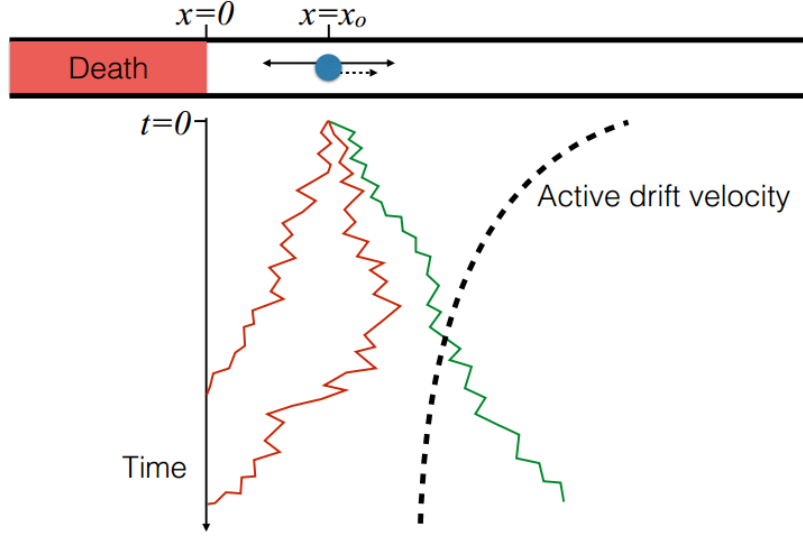


Figure 5.1: We consider a particle which is currently at position $x = x_0$ and which moves diffusively, but with the ability to burn energy and impose a drift velocity. At the origin is a stationary threat, which kills the particle upon contact (e.g. red trajectories). The problem is how to choose the drift velocity $v(t)$ so as to optimise a fitness functional balancing the cost of burning energy with the desire to survive (green trajectory).

select the actual drift velocity that it imposes on its diffusive motion.

The argument we make to derive this fitness functional is quite simplistic and does not amount to a full, rigorous application of the principle of FSM. Nevertheless, the general approach of making decisions in the present by modelling future trajectories remains the same, and the problem is interesting to study for its own sake. On top of this, whilst the model is extremely simple, it is not inconceivable that it could have some relevance to understanding threat avoidance by real microscopic swimming organisms, although this is certainly not our primary motivation. It also bears a close resemblance to a number of drift-diffusion “first-passage” problems, which have been studied in many different contexts[62].

5.1 Description of the model

The basic set up is shown schematically in figure 5.1. We consider an overdamped Brownian particle undergoing diffusion with a time-dependent drift velocity $v(t)$ over which it has complete control. The “threat” is stationary and placed at the origin, and

is such that it kills the particle immediately upon contact. This means that we can model it by incorporating an absorbing boundary condition. The particle is otherwise free to move anywhere in the region where $x > 0$. For simplicity, we make the choice to work in units where the diffusion constant $D = 1$, as well as defining $k_b T = 1$ (where T is the temperature and k_b is Boltzmann's constant). The evolution of the probability distribution $P(x, t)$ which describes the particle's position is then governed by the following drift-diffusion equation[63]:

$$\frac{\partial P(x, t)}{\partial t} = \frac{\partial^2 P(x, t)}{\partial x^2} - v(t) \frac{\partial P(x, t)}{\partial x} \quad (5.1)$$

We use this to model the probability distribution of the particle's position over future trajectories when a particular choice of $v(t)$ has been made. As such, we seek to solve this equation subject to the boundary conditions $P(x, 0) = \delta(x - x_0)$, since by definition the particle is making its decision starting from a known position, x_0 . We also want to impose $P(0, t) = 0 \forall t$ to account for the absorbing boundary condition, accounting for the presence of the threat.

An obvious quantity of interest is the probability of the particle dying after some time T . To calculate this we need to consider the *first-passage time probability*, i.e. the probability that the particle first reaches the origin at time t . For our setup this flux, which we call $j(t)$, is given by:

$$j(t) = \left. \frac{\partial P(x, t)}{\partial x} \right|_{x=0} \quad (5.2)$$

i.e. the rate at which probabilistic weight is absorbed onto the boundary. The probability of the particle having died at time T can then be written as:

$$P_d(T) = \int_0^T j(t) dt \quad (5.3)$$

5.1.1 Derivation of a fitness functional

Before making any attempt to try to calculate $P_d(T)$ we first consider how to choose the fitness functional that we wish to optimise. To do this, we seek an approximate expression representing the expected number of future paths which are available to the particle once the threat has stopped being "active". That is, we assume for whatever reason that the threat is only going to be present for a time T (which we will use to set our unit of time, such that $T = 1$), and that the particle anticipates that it will live

for a further time t_f if it survives the threat (on average). How many potential paths it then has available to it will depend on how much of its energy reserves it used up in escaping the threat. If, over its anticipated future lifetime, it is able to move with an average velocity \bar{v} , then it has the ability to travel a further distance $\bar{v}t_f$. Suppose we consider that there is an arbitrary minimum meaningful length scale ϵ (perhaps related to the size of the particle), then we can think of this (roughly) as being able to make $\frac{\bar{v}t_f}{\epsilon}$ steps, either to the left or the right. This corresponds to $2^{\frac{\bar{v}t_f}{\epsilon}}$ potential future trajectories. In this overdamped limit we can then say that the amount of energy needed to maintain an average speed of \bar{v} over this future time is given $E = \bar{v}^2 t_f$. As such, if the particle uses up an energy $E_{used} = \int_0^1 v(t)^2 dt$ in escaping the threat, then it has access to $\approx 2^{\frac{\sqrt{(E_{tot} - E_{used})t_f}}{\epsilon}}$ future trajectories, where here E_{tot} could be extended to include both energy reserves as well as the anticipated rate of energy collection in the future. On the other hand, if the particle is killed by the threat, it of course has zero future paths available to it, which means we can write the expected number of trajectories, Ω , as:

$$\Omega = (1 - P_d[1; v(t)]) 2^{\frac{\sqrt{(E_{tot} - E_{used}[v(t)])t_f}}{\epsilon}} \quad (5.4)$$

Clearly this is a simplified analysis as it doesn't, for example, account for the randomness in the future trajectories due to Brownian motion. However, since this is something the particle has no control over this is not necessarily a problem, and using this straightforward argument leads to an intuitive fitness functional which we can study analytically. Taking the logarithm of this and assuming that $E_{tot} \gg E_{used}$ so that we can expand the square root we find:

$$\log \Omega \approx \log(1 - P_d(1)) + \frac{\log(2)\sqrt{E_{tot}t_f}}{\epsilon} \left(1 - \frac{E_{used}[v(t)]}{E_{tot}}\right) \quad (5.5)$$

Maximising this with respect to the drift velocity $v(t)$ is equivalent to maximising the following functional:

$$J[v(t)] = \log(1 - P_d(1; v(t))) - \alpha \int_0^1 v(t)^2 dt \quad (5.6)$$

where here the second term is the total energy used multiplied by some “fitness parameter”, $\alpha = \frac{\log(2)\sqrt{t_f}}{\epsilon\sqrt{E_{tot}}}$. This is an intuitively sensible fitness to optimise, since it makes the particle sensitive to its energy usage, but also places importance on remaining alive. In fact, there is an especially strong cost placed on the whole ensemble of particles go-

ing extinct ($P_d(1) \rightarrow 1$), as this sends the fitness to minus infinity. Small values of α correspond to energy being “cheap” or plentiful, such that it is worthwhile to expend much energy if that will significantly improve the chance of survival. High values of α correspond to particles which need to be more frugal in their energy usage.

We can derive essentially the same expression (but with a different form for α) by instead considering the number of accessible *endpoints*, as opposed to the number of paths. Making the same assumptions as before there are $\approx \frac{2\bar{v}t_f}{\epsilon}$ accessible endpoints that can be reached, assuming that the particle survives. Therefore the expected number of these, Ω_s is given by:

$$\Omega_s = (1 - P_d(1)) \frac{2}{\epsilon} \sqrt{(E_{tot} - E_{used})t_f} \quad (5.7)$$

Taking the logarithm and again assuming $E_{tot} \gg E_{used}$ we find:

$$\log(\Omega_s) = \log(1 - P_d(1)) - \frac{E_{used}}{2E_{tot}} + \log\left(\frac{\sqrt{4E_{tot}t_f}}{\epsilon}\right) \quad (5.8)$$

Clearly maximising this is again equivalent to maximising equation 5.6, but with $\alpha = \frac{1}{2E_{tot}}$. Here α has no dependence on the seemingly arbitrary parameter ϵ , and so is perhaps easier to interpret.

5.2 Solving for the probability distribution in free space

Before we move on to trying to solve for the probability of the particle dying we first consider how to solve for the probability distribution of the particle in free space (i.e. without accounting for the absorbing boundary). That is, we look to solve equation 5.1 but using only the initial boundary condition $P(x, 0) = \delta(x - x_0)$. To do this we take the Fourier transform[64] of $P(x, t)$ in space:

$$\tilde{P}(k, t) = \int_{-\infty}^{\infty} P(x, t) e^{ikx} dx \quad (5.9)$$

Which has the following inversion:

$$P(x, t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \tilde{P}(k, t) e^{-ikx} dk \quad (5.10)$$

If we now consider multiplying equation 5.1 by e^{ikx} , and then integrating it from $-\infty$ to ∞ we have:

$$\int_{-\infty}^{\infty} e^{ikx} \frac{\partial P}{\partial t} dx = \int_{-\infty}^{\infty} e^{ikx} \frac{\partial^2 P}{\partial x^2} dx - v(t) \int_{-\infty}^{\infty} e^{ikx} \frac{\partial P}{\partial x} dx \quad (5.11)$$

We see that the term on the left is simply $\frac{d\tilde{P}}{dt}$ (as the derivative can be taken outside of the integral). The terms on the right can then be integrated by parts, which under the assumption that $P(x \rightarrow \pm\infty, t) = 0$ and $\frac{\partial P(x \rightarrow \pm\infty, t)}{\partial x} = 0$, leaves us with:

$$\frac{d\tilde{P}}{dt} = (-k^2 + ikv(t))\tilde{P} \quad (5.12)$$

This has now, in effect, converted the partial differential equation into an ordinary differential equation, which has the following solution:

$$\tilde{P}(k, t) = Ae^{-k^2 t + ik\bar{x}(t)} \quad (5.13)$$

where $\bar{x}(t) = x_0 + \int_0^t v(t') dt' \equiv x_0 + g(t)$. To satisfy the initial boundary condition we need to set $A = 1$, since we want $\tilde{P}(k, 0) = \int_{-\infty}^{\infty} \delta(x - x_0) e^{ikx} dx = e^{ikx_0}$. We can then take the inverse Fourier transform to obtain the solution for the probability distribution in free space, $P_{free}(x, t)$, finding that:

$$P_{free}(x, t) = \frac{1}{\sqrt{4\pi t}} e^{-\frac{(x - \bar{x}(t))^2}{4t}} \quad (5.14)$$

As we shall see in the next section, having this solution will prove to be useful in solving for the probability of the particle dying when we include the absorbing boundary.

5.3 Solving the first passage problem

To optimise the fitness functional we introduced in 5.1.1 we need to be able to calculate the probability of the particle dying, $P_d(1)$. This is obtained by integrating $j(t)$, probability flux out to the absorbing boundary. These types of first-passage problems, in which we seek to calculate the probability that a stochastic process first reaches a specified site at a given time, appear in many different contexts and have been widely studied[62]. In fact, the solution for our problem in the case where $v(t) = 0$ or $v(t) = v$ (a constant) is known and can be obtained relatively straightforwardly using the method of images. Starting with the simplest case where $v(t) = 0$ such that the system is only

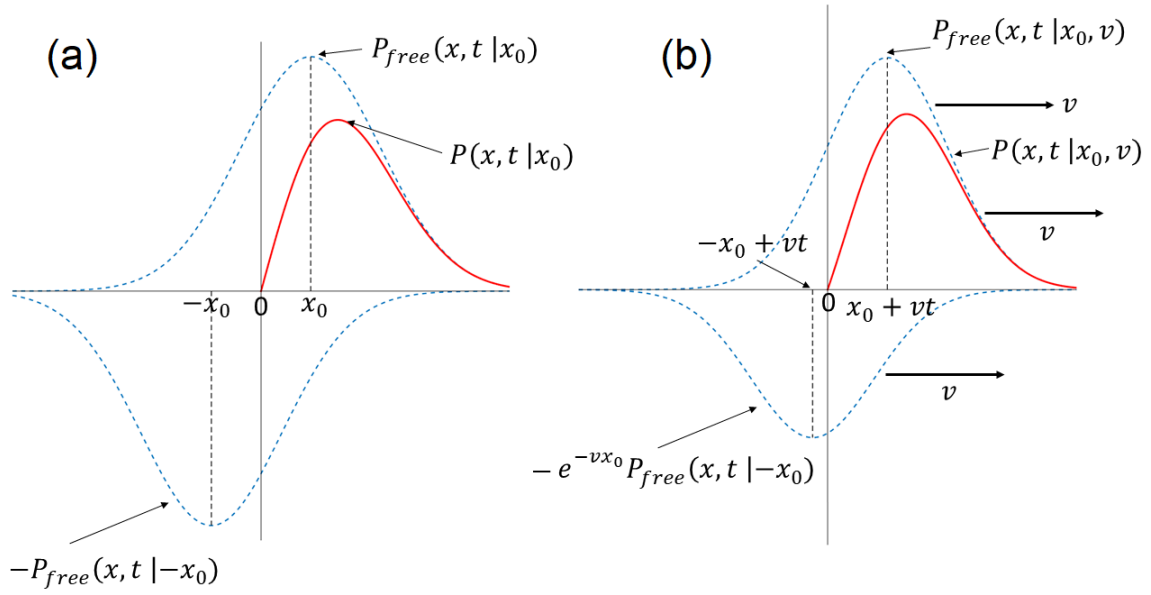


Figure 5.2: (a) Method of images in the case where $v = 0$. To obtain the probability distribution for $x > 0$ in the presence of an absorbing boundary, we consider superimposing a negative image particle starting at $-x_0$ with the solution for the particle starting at x_0 . (b) When a constant drift velocity is included a similar approach can be taken. Both the “real” particle and the image particle move with drift velocity v , but the image particle is scaled so as to ensure $P(0, t) = 0$.

undergoing diffusion, the idea here is consider the solution in free space for a particle starting at $x_0 > 0$ that we derived in the previous section. We then superimpose on top of this a negative solution from an “image” particle or an “anti-particle” which starts at $-x_0$. That is, we write:

$$P(x, t) = \frac{1}{\sqrt{4\pi t}} \left[e^{-\frac{(x-x_0)^2}{4t}} - e^{-\frac{(x+x_0)^2}{4t}} \right] \quad (5.15)$$

Clearly this satisfies the diffusion equation, and evidently $P(0, t) = 0$. Furthermore, as we are only interested in $x > 0$, where it satisfies the correct initial condition, this is exactly the solution that we seek. The first passage probability density $j(t)$ can then be obtained from equation 5.2. The idea behind this method is shown schematically in figure 5.2(a). In the case that we have a constant drift velocity, i.e. $v(t) = v$, the naive approach may be to try and do the same thing, including a negative image particle starting at $-x_0$ which moves at $-v$. Whilst this would certainly impose the correct

boundary condition, the solution would no longer satisfy the drift-diffusion equation since the negative image is moving with the wrong drift velocity, and so this does not work. We can, however, use a negative image particle moving in the same direction as the “real” particle but with a different weighting. That is, we consider:

$$P(x, t) = \frac{1}{\sqrt{4\pi t}} \left[e^{-\frac{(x-x_0-vt)^2}{4t}} - we^{-\frac{(x+x_0-vt)^2}{4t}} \right] \quad (5.16)$$

Provided that w is a constant, this still satisfies the drift-diffusion equation. It is easy to show that if we take $w = e^{-vx_0}$ then this leads to the correct boundary condition $P(0, t) = 0$ being satisfied for all times, and so again gives us the solution we seek. Following this through we can obtain the first passage probability density:

$$j(t) = \frac{x_0}{\sqrt{4\pi t^3}} e^{-\frac{(x_0+vt)^2}{4t}} \quad (5.17)$$

Despite this being a nice extension of the method of images to biased diffusion, unfortunately it also demonstrates how this method breaks down when we consider $v(t)$ to be an arbitrary, time-varying function. In this case we would need to weight the negative image particle by an amount $e^{\frac{\int_0^t v(t')dt'}{4t}}$, which is no longer a constant, and as such the resultant solution which we obtain will not satisfy equation 5.1. As far as we are aware, there is no known analytic solution for $P(x, t)$ when we have an absorbing boundary in the case that $v(t)$ is arbitrary. In this section we take a different approach and are able to derive an exact integral equation for $j(t)$ directly, without having to first obtain $P(x, t)$.

To do this we consider the propagator of the equation in free space, $G(x, x', t, t')$, which should satisfy the following three properties:

$$\begin{aligned} P_{free}(x, t) &= \int_{-\infty}^{\infty} G(x, x', t, 0) \delta(x' - x_0) dx' \quad t > 0 \\ P_{free}(x, t) &= \int_{-\infty}^{\infty} G(x, x', t, t') P_{free}(x', t') dx' \quad t > t' \\ G(x, x'', t_2, t_0) &= \int_{-\infty}^{\infty} G(x, x', t_2, t_1) G(x', x'', t_1, t_0) dx' \quad t_2 > t_1 > t_0 \end{aligned} \quad (5.18)$$

as well as itself being a solution to equation 5.1. Intuitively, this is a function which acts on some initial condition distribution and propagates it forwards to give the correct solution of equation 5.1 at a later time. It is easy to verify that the function which

satisfies these properties for equation 5.1 in free space is:

$$G(x, x', t, t') = \frac{1}{\sqrt{4\pi(t-t')}} e^{-\frac{(x-x'-(\bar{x}(t)-\bar{x}(t'))))^2}{4(t-t')}} \quad (5.19)$$

If now we consider solving the drift-diffusion equation in free space with the initial delta function source $\delta(x - x_0)$, but also in the presence of an additional source/sink of probability which injects/absorbs probability into the system with a rate $s(x, t)$, then we can write the total resultant probability distribution as:

$$P(x, t) = P_{free}(x, t) + \int_{-\infty}^{\infty} dx' \int_0^t dt' G(x, x', t, t') s(x', t') \quad (5.20)$$

We now make the following observation: an absorbing boundary is, physically, really nothing more than a probability sink. In particular, an absorbing boundary placed at $x = 0$ is a probability sink that removes probability from the system with rate $j(t)$, in such a way as to ensure that $P(0, t) = 0 \forall t$. As such, if we substitute $s(x', t') = j(t')\delta(x')$ into the above equation, we can write an equation that must be satisfied by the probability distribution at any given x and t in the presence of the absorbing boundary:

$$P(x, t) = P_{free}(x, t) - \int_0^t G(x, 0, t, t') j(t') dt' \quad (5.21)$$

But specifically, we know that if we evaluate this at $x = 0$ it must equal zero, i.e.:

$$P(0, t) = 0 = P_{free}(0, t) - \int_0^t G(0, 0, t, t') j(t') dt' \quad (5.22)$$

This gives the following integral equation that $j(t)$ must satisfy:

$$\boxed{\frac{1}{\sqrt{t}} e^{-\frac{\bar{x}(t)^2}{4t}} = \int_0^t \frac{j(t')}{\sqrt{t-t'}} e^{-\frac{(\bar{x}(t)-\bar{x}(t'))^2}{4(t-t')}} dt'} \quad (5.23)$$

As a check, we show we can use this integral equation to recover the same results as we obtained using the method of images when $v(t) = v$. We start by considering $v = 0$, such that the integral equation gives us:

$$\frac{e^{-\frac{x_0^2}{4t}}}{\sqrt{t}} = \int_0^t \frac{j(t')}{\sqrt{t-t'}} dt' \quad (5.24)$$

This is an Abel integral equation[65], and can be solved exactly. In general, if we have

an equation of the form:

$$\int_a^x \frac{\phi(x')}{|x - x'|^\alpha} dx' = f(x), \quad a \leq x \leq b \quad (5.25)$$

where $f(x)$ is a known function and $0 < \alpha < 1$, then the unknown function $\phi(x)$ satisfies:

$$\phi(x) = \frac{\sin(\alpha\pi)}{\pi} \frac{d}{dx} \left[\int_a^x \frac{f(x')}{(x - x')^{1-\alpha}} dx' \right] \quad (5.26)$$

In our situation this gives:

$$j(t) = \frac{1}{\pi} \frac{d}{dt} \left[\int_0^t \frac{1}{\sqrt{t'(t-t')}} e^{-\frac{x_0^2}{4t'}} dt' \right] \quad (5.27)$$

If we define the integral in the above equation as $I(t)$, then we note that $I(t) = \int_0^t F(t')G(t-t')dt'$, where $F(t) = \frac{1}{\sqrt{t}}e^{-\frac{x_0^2}{4t}}$ and $G(t) = \frac{1}{\sqrt{t}}$. We can then consider the Laplace transform of $I(t)$ to be $\hat{I}(s)$, which by the convolution theorem is equal to $\hat{I}(s) = \hat{F}(s)\hat{G}(s)$. Fortunately, the Laplace transforms of both $F(t)$ and $G(t)$ are known and given by $\hat{F}(s) = \sqrt{\frac{\pi}{s}}$ and $\hat{G}(s) = \sqrt{\frac{\pi}{s}}e^{-x_0\sqrt{s}}$ respectively. This means that $\hat{I}(s) = \frac{\pi}{s}e^{-x_0\sqrt{s}}$, which can be inverted to give $I(t) = \pi \operatorname{erfc}\left[\frac{x_0}{\sqrt{4t}}\right]$, where $\operatorname{erfc}[\cdot]$ is the complementary error function. It is then trivial to differentiate this to obtain $j(t) = \frac{x_0}{\sqrt{4\pi t^3}}e^{-\frac{x_0^2}{4t}}$, which is exactly what we obtained from the method of images (see equation 5.17). Generalising this to the case where $v(t)$ is a constant, equation 5.23 now gives us:

$$\frac{1}{\sqrt{t}}e^{-\frac{(x_0+vt)^2}{4t}} = e^{-\frac{v^2 t}{4}} \int_0^t \frac{j(t')}{\sqrt{t-t'}} e^{\frac{v^2 t'}{4}} dt' \quad (5.28)$$

Writing $J(t) = j(t)e^{\frac{v^2 t}{4}}$, it is clear we again have an Abel integral equation which can be solved in exactly the same manner. Following this through it is then relatively straightforward to show that we get $j(t) = \frac{x_0}{\sqrt{4\pi t^3}}e^{-\frac{(x_0+vt)^2}{4t}}$, again replicating the result obtained using the method of images.

As a further verification of the validity of this approach we consider computing numerically the full probability distribution $P(x, t)$ using equation 5.21, in the case where $v(t)$ is a constant. We can then compare this to the known exact solution obtained from the method of images. An example comparing the two approaches is shown in figure 5.3 for $v = 0.5$ and $x_0 = 1.2$, where we compute the distribution over x -values at time $t = 3$. We see that in the region we are interested in, $x > 0$, the agreement is almost

perfect, but furthermore we see that it also gives us the “correct” result in the region where $x < 0$. Clearly here we would expect that $P(x, t) = 0$ since no probability mass is able to pass the absorbing boundary, which is exactly what we obtain.

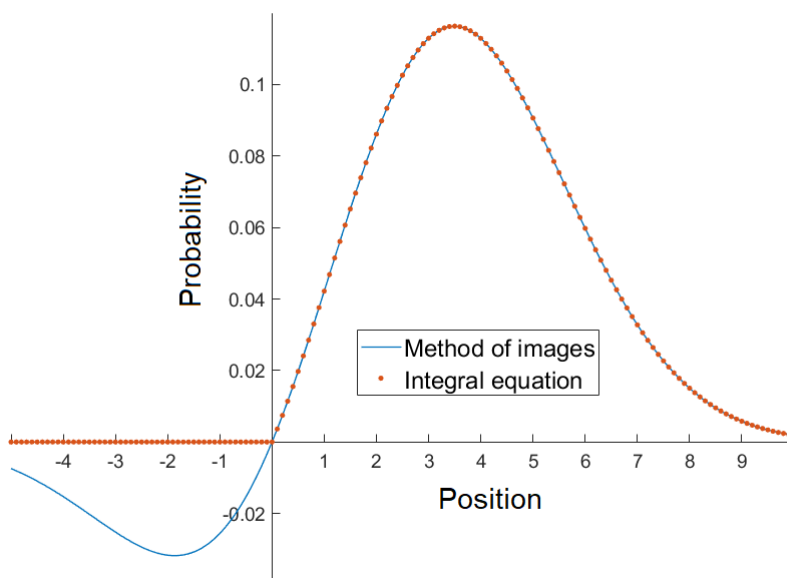


Figure 5.3: Example showing $P(x, t)$ in the case where $v(t)$ is a constant, in the presence of an absorbing boundary. We plot $P(x, t)$ for $v = 0.5$ and $x_0 = 1.2$ at a given time $t = 3$ using two approaches - firstly by using the exact solution given by the method of images (equation 5.16, and then secondly by evaluating numerically equation 5.23 (using the known form for $j(t)$). We see that the agreement in the region of interest, $x > 0$, is essentially exact.

So at this point we have an exact integral equation for the first-passage probability density $j(t)$ in the case that $v(t)$ is completely arbitrary. Unfortunately we are not aware of a general way to solve this analytically, however it is possible to obtain a numerical solution with reasonable efficiency[66] (and in a way which is significantly easier than trying to directly obtain a numerical solution to the partial differential equation with the given boundary conditions). We shall discuss these numerical approaches in detail later as they form the basis for the method we use to numerically solve for the optimal drift velocity.

Finally, it is worth mentioning that this approach can be generalised fairly straight-

forwardly to situations in which the absorbing boundary is not stationary, as well as cases where we include a time-dependent diffusion constant in the equation we are trying to solve. Let us consider the first passage problem of a system described by the following PDE:

$$\frac{\partial P(x, t)}{\partial t} = \sigma(t) \frac{\partial^2 P(x, t)}{\partial x^2} - v(t) \frac{\partial P(x, t)}{\partial x} \quad (5.29)$$

with an absorbing boundary which moves so that at time t it is at position $B(t)$. If we define $D(t) = \int_0^t \sigma(t') dt'$ then we can write the propagator of this equation in free space as:

$$G(x, x', t, t') = \frac{1}{\sqrt{4\pi(D(t) - D(t'))}} e^{-\frac{(x-x' - (\bar{x}(t) - \bar{x}(t')))^2}{4(D(t) - D(t'))}} \quad (5.30)$$

and the free space solution starting from a delta function initial condition at x_0 is given by:

$$P_{free} = \frac{1}{\sqrt{4\pi D(t)}} e^{-\frac{(x - \bar{x}(t))^2}{4D(t)}} \quad (5.31)$$

To solve in the presence of the absorbing boundary we then consider a probability sink which moves, $s(x', t') = -j(t')\delta(x' - B(t'))$. We then make the same step as before, except now we substitute in $x = B(t)$ instead of $x = 0$. This gives us the following integral equation:

$$P_{free}(B(t), t) = \int_0^t G(B(t), B(t'), t, t') j(t') dt' \quad (5.32)$$

5.4 Approximate analytical solution when diffusion is dominant

Before we present a numerical approach to solve for the optimal drift velocity we first look for an analytic solution which is valid under certain approximations. This is useful to get some intuition into the solution, and it also turns out to provide a good initial guess for obtaining a full numerical solution. Let us consider the limit where diffusion dominates over drift, such that we can approximate $\frac{(\bar{x}(t) - \bar{x}(t'))^2}{4(t - t')} \ll 1$. This allows us to drop the exponential term on the right hand side of equation 5.23. If we also make a Taylor expansion to first order in the left-hand side of the equation then we can write:

$$\frac{e^{-\frac{x_0^2}{4t}}}{\sqrt{t}} \left(1 - \frac{x_0}{2t} g(t)\right) \approx \int_0^t \frac{j(t')}{\sqrt{t - t'}} dt' \quad (5.33)$$

(where $g(t) = \int_0^t v(t')dt'$). This is again an Abel integral equation with the following solution:

$$j(t) = \frac{1}{\pi} \frac{d}{dt} \left[\int_0^t \frac{e^{-\frac{x_0^2}{4t'}}}{\sqrt{t'(t-t')}} \left(1 - \frac{x_0 g(t')}{2t'}\right) dt' \right] \quad (5.34)$$

The fact that this is left as a derivative is actually extremely convenient since what we really want is the probability of dying, which we now obtain immediately:

$$P_d(T) = \frac{1}{\pi} \int_0^T \frac{e^{-\frac{x_0^2}{4t}}}{\sqrt{t(T-t)}} \left(1 - \frac{x_0 g(t)}{2t}\right) dt \quad (5.35)$$

We note that the first term in this integral has already been evaluated in equation 5.27, and simply gives the probability of the agent dying when there is zero drift velocity. The second term can then be integrated by parts such that we are left with:

$$P_d(T) = \operatorname{erfc} \left[\frac{x_0}{\sqrt{4T}} \right] - \sqrt{\frac{1}{\pi T}} e^{-\frac{x_0^2}{4T}} \int_0^T v(t) \operatorname{erf} \left[\sqrt{\frac{x_0^2(T-t)}{4Tt}} \right] dt \quad (5.36)$$

This is simply the probability of dying with no diffusion along with a first-order correction, valid in the limit that $v(t)$ is small enough.

We can make use of this approximate expression for P_d to derive an expression for the optimal drift velocity of the particle, again valid under the assumption that diffusion is dominant. Let us write an arbitrary drift velocity as $v(t) = v^*(t) + \epsilon \delta v(t)$, where $v^*(t)$ is the optimal drift velocity and ϵ is simply a real parameter. This allows us to write the fitness functional in equation 5.6 (making use of the approximation in equation 5.36 and setting $T = 1$) as:

$$J[\epsilon] = \log \left[1 - \operatorname{erfc} \left[\frac{x_0}{2} \right] + \sqrt{\frac{1}{\pi}} e^{-\frac{x_0^2}{4}} \int_0^1 (v^*(t) + \epsilon \delta v(t)) \operatorname{erf} \left[\sqrt{\frac{x_0^2(1-t)}{4t}} \right] dt \right] - \alpha \int_0^1 (v^*(t) + \epsilon \delta v(t))^2 dt \quad (5.37)$$

This depends only on ϵ , and so the condition that this is an extremum is given by:

$$\left. \frac{dJ}{d\epsilon} \right|_{\epsilon=0} = 0 \quad (5.38)$$

i.e.:

$$\int_0^1 \left[\frac{\exp\left(-\frac{x_0^2}{4}\right)}{\sqrt{\pi}(1 - P_d[1; v^*(t)])} \operatorname{erf}\left[\sqrt{\frac{x_0^2(1-t)}{4t}}\right] - 2\alpha v^*(t) \right] \delta v(t) dt = 0 \quad (5.39)$$

But since this is true for arbitrary $\delta v(t)$, it must be the case that:

$$v^*(t)(1 - P_d[1; v^*(t)]) = \frac{e^{-\frac{x_0^2}{4}}}{\alpha\sqrt{4\pi}} \operatorname{erf}\left[\sqrt{\frac{x_0^2(1-t)}{4t}}\right] \quad (5.40)$$

This gives us an equation that must be satisfied by the optimal drift velocity, which from now on we write as $v(t)$ rather than $v^*(t)$, for notational convenience. To solve this equation we consider two infinite dimensional vectors \vec{v} and \vec{a} with components given by $v(t)$ and $\operatorname{erf}\left[\sqrt{\frac{x_0^2(1-t)}{4t}}\right]$ respectively (for all values of $t \in [0, 1]$). We define the inner product between them in the following way:

$$\vec{a} \cdot \vec{v} = \int_0^1 v(t) \operatorname{erf}\left[\sqrt{\frac{x_0^2(1-t)}{4t}}\right] dt \quad (5.41)$$

This allows us to re-write equation 5.40 in vector form in the following way:

$$(2\alpha(P_0 + C_0(\vec{a} \cdot \vec{v})) \vec{v} = C_0 \vec{a} \quad (5.42)$$

Where $P_0 = \operatorname{erf}(x_0/2)$ and $C_0 = \frac{1}{\sqrt{\pi}} e^{-\frac{x_0^2}{4}}$. This makes it clear that $v(t)$ takes the following form:

$$v(t) = \beta \operatorname{erf}\left[\sqrt{\frac{x_0^2(1-t)}{4t}}\right] \quad (5.43)$$

from which it is straightforward to show that β is given by:

$$\beta = \frac{\sqrt{2(\vec{a} \cdot \vec{a})C_0^2\alpha + P_0^2\alpha^2} - P_0\alpha}{2(\vec{a} \cdot \vec{a})C_0\alpha} \quad (5.44)$$

The first thing to note here is that the largest value of $v(t)$ is always at $t = 0$. Intuitively this makes some sense as it tells us that the optimal strategy is to act early, rather than waiting for diffusion to blur the particle's position. Still, *a priori* this wasn't necessarily obvious. We can also see that the form of the *time dependence* of the optimal drift is influenced only by x_0 , with no dependence on the fitness parameter α . The only place where α enters under this approximation is in the constant term, β . We shall see later on

that this is no longer true once we obtain a full numerical solution for the optimal drift velocity, without approximations. Under this approximation we also find that for small α , $\beta \sim \alpha^{-1/2}$. This means that as $\alpha \rightarrow \infty$, energy becomes extremely expensive and so $\beta \rightarrow 0$, as it should. For small x_0 (when the particle is near to the threat), as well as for early times, the leading order behaviour is $v(t) \sim \frac{1}{\sqrt{t}}$. We can make an argument that this is what is required for the drift to “match” diffusion in a scaling sense. We do this as follows: consider the time taken for the root mean-squared displacement of the particle due to diffusion to be equal to x_0 , which is given by $\tau = \frac{x_0^2}{2}$. To “match” this we want a drift velocity that moves us a distance x_0 in time τ . That is, we want:

$$\int_0^\tau v(t)dt = x_0 = \sqrt{2\tau} \quad (5.45)$$

which is satisfied if $v(t) = \sqrt{\frac{2}{t}}$. Of course, finding that we get the kind of time-dependence that “matches” diffusion when we have explicitly assumed that diffusion dominates over drift is not particularly convincing evidence that the true optimal drift will realise this — nevertheless later on we shall find that actually this functional form of the time-dependence closely approximates the full numerical solution. Figure 5.4(a) shows how the functional form of this approximate optimal drift velocity varies with x_0 . The smaller, inset plot also shows the same results on a log-log scale where it is much easier to see how close the form is to $t^{-1/2}$ when x_0 is made very small. Figure 5.4(b) shows how the constant β varies with the fitness parameter (again on a log-log scale), and a line showing an $\alpha^{-1/2}$ power-law is also included for comparison.

5.5 Full numerical solution for the optimal drift velocity

The approximate analytic solution we have obtained has some interesting features which are all reasonably intuitive. Nevertheless, the approximation that diffusion dominates over drift is potentially very restrictive, and in reality the most interesting regime may be the one where the drift and diffusion contributions are comparable. Clearly it is therefore desirable to develop a method which can be used to obtain a numerical solution for the optimal drift velocity without having to make any approximations. How to go about doing this is the question we address in this section.

The problem we have is to try and optimise the following fitness functional with re-

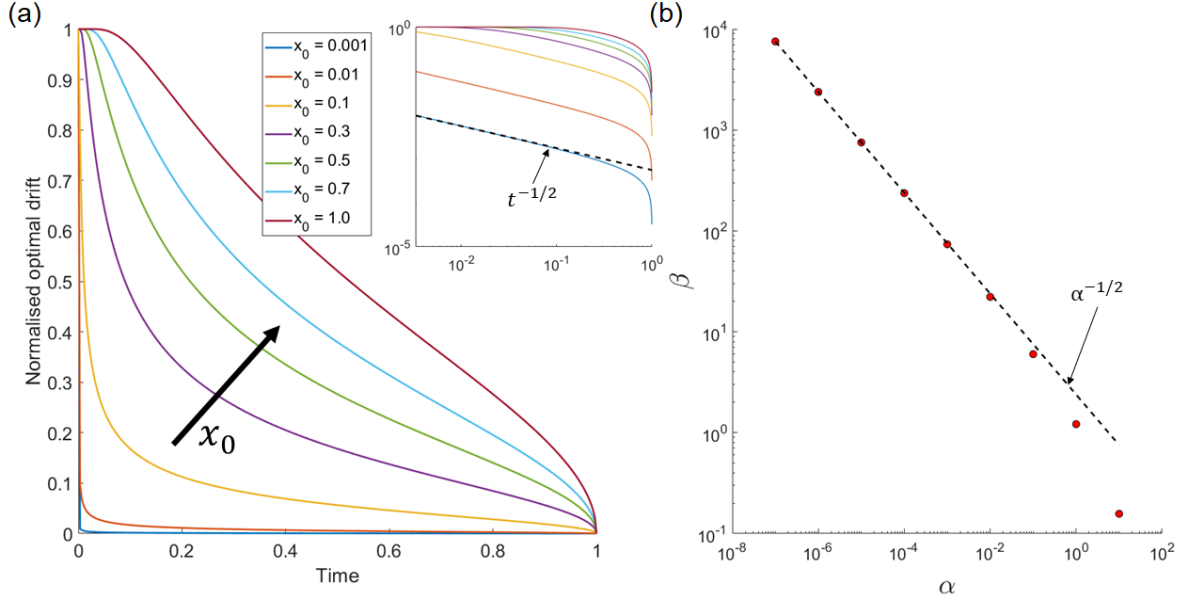


Figure 5.4: (a) Functional form of the predicted optimal drift velocity (equation 5.43) over the future time interval being modelled, for different values of the particle’s initial starting point x_0 . (b) The initial value of the optimal drift velocity (i.e. at $t = 0$) shown as a function of the fitness parameter α . In this example x_0 is fixed at $x_0 = 0.2$.

spect to the optimal drift velocity over the modelled future time interval:

$$J[v(t)] = \log(1 - \int_0^1 j(t; v(t)) dt) - \alpha \int_0^1 v(t)^2 dt \quad (5.46)$$

The main issue we have here is that $j(t)$ is something which we do not have a closed analytic expression for (we only have the integral equation 5.23), which prevents us from being able to directly apply techniques from the calculus of variations or optimal control theory[67]. To get around this, we instead think of this fitness functional as depending on two “independent” functions $j(t)$ and $\bar{x}(t)$ (remembering that $v(t) = \bar{x}'(t)$), and then requiring that the integral equation relating the two holds as a constraint. That is, we introduce a time-dependent Lagrange multiplier $\lambda(t)$ and we maximise:

$$J[\bar{x}(t), j(t)] = \log \left[1 - \int_0^1 j(t) dt \right] - \int_0^1 \left[\alpha \bar{x}'(t)^2 + \lambda(t) \left(\frac{\exp \left(-\frac{\bar{x}(t)^2}{4t} \right)}{\sqrt{t}} - \int_0^t \frac{j(t')}{\sqrt{t-t'}} \exp \left(-\frac{(\bar{x}(t) - \bar{x}(t'))^2}{4(t-t')} \right) dt' \right) \right] dt \quad (5.47)$$

This means that we need to take the functional derivatives of J with respect to both $j(t)$ and $\bar{x}(t)$. This will then give us two equations, which taken together with the integral equation constraint will provide 3 equations for the 3 unknowns $j(t)$, $\bar{x}(t)$ and $\lambda(t)$. To do this, we write $j(t) = j^*(t) + \epsilon_1 \delta j(t)$ and $\bar{x}(t) = \bar{x}^*(t) + \epsilon_2 \delta \bar{x}(t)$ (where $j^*(t)$ and $\bar{x}^*(t)$ are the optimal functions), and require that:

$$\left. \frac{\partial J}{\partial \epsilon_1} \right|_{\epsilon_1, \epsilon_2=0} = \left. \frac{\partial J}{\partial \epsilon_2} \right|_{\epsilon_1, \epsilon_2=0} = 0 \quad (5.48)$$

To start with let us consider the first of these, which gives:

$$\left. \frac{\partial J}{\partial \epsilon_1} \right|_{\epsilon_1, \epsilon_2=0} = \frac{1}{1 - P_d(j^*)} \int_0^1 \delta j(t) dt - \int_0^1 dt \lambda(t) \int_0^t dt' \frac{\delta j(t') \exp\left(-\frac{(\bar{x}(t) - \bar{x}(t'))^2}{4(t-t')}\right)}{\sqrt{t-t'}} = 0 \quad (5.49)$$

We can cast this into a more useful form by noting that for any arbitrary function $g(t, t')$ the following identity is true:

$$\int_0^T dt \int_0^t dt' g(t, t') = \int_0^T dt \int_t^T dt' g(t', t) \quad (5.50)$$

This means that we can re-write equation 5.49 as follows, such that the arbitrary $\delta j(t)$ can be factored out:

$$\left. \frac{\partial J}{\partial \epsilon_1} \right|_{\epsilon_1, \epsilon_2=0} = \frac{1}{1 - P_d(j^*)} \int_0^1 \delta j(t) dt - \int_0^1 dt \delta j(t) \int_t^1 dt' \frac{\lambda(t') \exp\left(-\frac{(\bar{x}(t') - \bar{x}(t))^2}{4(t'-t)}\right)}{\sqrt{t'-t}} = 0 \quad (5.51)$$

Since this is true for arbitrary $\delta j(t)$, we obtain an integral equation which must be satisfied by the optimal functions:

$$\boxed{\frac{1}{1 - P_d(j^*)} = \int_t^1 \frac{\lambda(t')}{\sqrt{t'-t}} \exp\left(-\frac{(\bar{x}(t') - \bar{x}(t))^2}{4(t'-t)}\right) dt'} \quad (5.52)$$

The second condition $\left. \frac{\partial J}{\partial \epsilon_2} \right|_{\epsilon_1, \epsilon_2=0}$ is a bit trickier, but essentially the procedure we follow is the same. Looking at the term $\alpha \int_0^1 \bar{x}'(t)^2 dt = \alpha \int_0^1 (\bar{x}'^*(t) + \epsilon_2 \delta \bar{x}'(t))^2 dt$ we can differentiate with respect to ϵ_2 and integrate by parts (restricting $\delta \bar{x}(t)$ to the set of functions which go to zero at $t = 0, 1$) so that we are left with $-\alpha \int_0^1 \bar{x}''^*(t) \delta \bar{x}(t) dt$. We

then make use of the following:

$$\begin{aligned} \frac{\partial}{\partial \epsilon_2} \left[\exp \left(-\frac{(\bar{x}^*(t) + \epsilon_2 \delta \bar{x}(t) - \bar{x}^*(t') - \epsilon_2 \delta \bar{x}^*(t'))^2}{4(t-t')} \right) \right]_{\epsilon_1, \epsilon_2=0} = \\ \frac{-u}{2(t-t')} \exp \left(-\frac{u^2}{4(t-t')} \right) (\delta \bar{x}(t) - \delta \bar{x}(t')) \end{aligned} \quad (5.53)$$

where $u = \bar{x}^*(t) - \bar{x}^*(t')$. The identity given in equation 5.50 can then be used again such that we can factor out $\delta \bar{x}(t)$, giving us another equation that must be satisfied by the optimal functions:

$$\boxed{\begin{aligned} \frac{d^2 \bar{x}(t)}{dt^2} = & \frac{-\lambda(t) \bar{x}(t)}{4\alpha t^{3/2}} e^{-\frac{\bar{x}(t)^2}{4t}} + \frac{\lambda(t)}{4\alpha} \int_0^t \frac{j(t')(\bar{x}(t) - \bar{x}(t'))}{(t-t')^{3/2}} \exp \left(-\frac{(\bar{x}(t) - \bar{x}(t'))^2}{4(t-t')} \right) dt' \\ & - \frac{j(t)}{4\alpha} \int_t^1 \frac{\lambda(t')(\bar{x}(t') - \bar{x}(t))}{(t'-t)^{3/2}} \exp \left(-\frac{(\bar{x}(t') - \bar{x}(t))^2}{4(t'-t)} \right) dt' \end{aligned}} \quad (5.54)$$

At this stage we are in a position where we have 3 coupled integro-differential equations to solve for $\bar{x}(t)$, $j(t)$ and $\lambda(t)$ (with the boundary conditions $\bar{x}(0) = x_0$ and $\bar{x}'(1) = 0$, i.e. the drift velocity goes to zero at the end when the threat stops being active). These are somewhat ugly, and the fact that they are nonlinear and weakly singular (as $(t-t')^{-1/2}$ diverges as $t \rightarrow t'$) makes them particularly difficult to solve. Whilst there exists a vast literature[68] for the numerical solution of nonlinear ordinary differential equations where methods exist that can be directly applied to a new system with relative ease, our experience has been that general techniques for tackling nonlinear integro-differential equations are much less developed. Some potential candidate methods worth mentioning include the “Compact Finite Difference Method”[69], the “Variational Iteration Method”[70] and the “Homotopy Analysis Method”[71]. However, for all of these we encountered problems, either due to the singularities in the integrands or because of the unusual boundary conditions that we have on $\bar{x}(t)$ (where we specify an initial value and a final value on its derivative). With more time it may have been possible to adapt one of these methods to obtain a good numerical solution, however in the end we came up with our own method based on modifying the “trapezoidal product integration” method, used to numerically solve generalised Abel equations[66] such as equation 5.23. By combining this with Newton’s method for solving nonlinear systems of equations[72] we are able

to get a method which can be used iteratively to solve for an optimal $\bar{x}(t)$, and hence $v(t)$.

Our method can be understood as follows. We start with an initial guess for $\bar{x}(t)$ which we can obtain, for example, from the approximate analytic solution found in the previous section. We then use equation 5.23 where we substitute this initial guess for $\bar{x}(t)$ and solve it numerically to obtain a first guess for $j(t)$. These two guesses can then be used in solving equation 5.52 to obtain an estimate for $\lambda(t)$. Finally, we use the new guesses we've obtained for $j(t)$ and $\lambda(t)$ to solve equation 5.54 for a new value of $\bar{x}(t)$. This procedure is then iterated until the functions converge to the correct solution, within some small tolerance.

To do this, the first thing we need to do is discretise the time interval $[0, 1]$, such that we solve for the functions $j(t)$, $\bar{x}(t)$ and $\lambda(t)$ only at the discrete set of points $t_i = i\Delta t$, where $i = \{0, 1, \dots, I = \frac{1}{\Delta t}\}$ ($\frac{1}{\Delta t}$ is assumed to be an integer). Now let us take our initial guess for $\bar{x}(t)$ and consider equation 5.23. If we define $h(t) = \frac{1}{\sqrt{t}}e^{-\frac{\bar{x}(t)^2}{4t}}$ and $K(t, t') = e^{-\frac{(\bar{x}(t) - \bar{x}(t'))^2}{4(t - t')}}$, this allows us to write down a series of equations that must be satisfied for each value of i :

$$h(t_i) \equiv h_i = \int_0^{t_i} \frac{j(t')}{\sqrt{t_i - t'}} K(t_i, t') dt' \quad (5.55)$$

The natural thing to do now is to try and approximate this integral using values of the functions at the discrete times we are considering. The normal approach for doing this would be to use (for example) the trapezium rule or Simpson's rule[73] to obtain such an approximation, however the situation here is complicated slightly since we have a singularity in the integrand as $t' \rightarrow t_i$. The trapezoidal product integration method proceeds by replacing $j(t')K(t_i, t')$ in each interval $t_m \leq t' \leq t_{m+1}$, ($m = 0, \dots, i - 1$) by the linear interpolation of the function which passes through its values at t_m and t_{m+1} , i.e.:

$$K(t_i, t')j(t') \rightarrow \frac{(t_{m+1} - t')K_{i,m}j_m + (t' - t_m)K_{i,m+1}j_{m+1}}{\Delta t} \equiv M_{i,m}(t') \quad (5.56)$$

where we define $K_{i,m} = K(t_i, t_m)$ and $j_i = j(t_i)$. That is, we have replaced the function $j(t')K(t_i, t')$ by a piece-wise linear function passing through the actual function values

at each discrete point in time being considered. As such, we can write equation 5.55 as:

$$h_i = \sum_{m=0}^{i-1} \int_{t_m}^{t_{m+1}} \frac{M_{i,m}(t')}{\sqrt{t_i - t'}} dt' \quad (5.57)$$

The advantage of doing this is that we can now actually evaluate these integrals analytically. If we make the substitutions $l = i - m$ and $\tau = t_i - t'$ we find:

$$h_i = \sum_{m=0}^{i-1} \int_{(l-1)\Delta t}^{l\Delta t} \frac{[(\tau - (l-1)\Delta t)K_{i,m}j_m + (l\Delta t - \tau)K_{i,m+1}j_{m+1}]}{\Delta t \tau^{1/2}} d\tau \quad (5.58)$$

Or by grouping terms together, more simply:

$$h_i = \sum_{m=1}^i j_m K_{i,m} W_{i-m}, \quad i \in \{1, 2, \dots, I\} \quad (5.59)$$

where we make use of the fact that $j_0 = 0$ and define:

$$\begin{aligned} W_0 &= \int_0^{\Delta t} \frac{(\Delta t - \tau)}{\Delta t} \tau^{1/2} d\tau = \frac{4}{3} \Delta t^{1/2} \\ W_l &= \int_{(l-1)\Delta t}^{l\Delta t} \frac{[\tau - (l-1)\Delta t]}{\Delta t \tau^{1/2}} d\tau + \int_{l\Delta t}^{(l+1)\Delta t} \frac{[(l+1)\Delta t - \tau]}{\Delta t \tau^{1/2}} d\tau \\ &= \frac{4}{3} \Delta t^{1/2} \left[(l-1)^{3/2} - 2l^{3/2} + (l+1)^{3/2} \right] \end{aligned} \quad (5.60)$$

This is now just a simple system of lower triangular linear equations and so can easily be solved using forward substitution[74] to get the new estimate for the optimal $j(t)$.

We can take an extremely similar approach for equation 5.52 where we are now solving for the unknown function $\lambda(t)$, although here we have to be a bit more careful as it turns out that this has a singularity in it. To see that this must be the case, let us make the substitution $t'' = 1 - t'$ and $\tilde{t} = 1 - t$, whilst defining $\theta(t') = \lambda(1 - t')$. The equation can then be rewritten as:

$$a = \int_0^{\tilde{t}} \frac{\theta(t'')}{\sqrt{\tilde{t} - t''}} e^{-\frac{(\bar{x}(1-t'') - \bar{x}(1-\tilde{t}))^2}{4(\tilde{t} - t'')}} dt'' \quad (5.61)$$

where $a = \frac{1}{1-P_d}$ is taken to be a constant. For simplicity, let's initially consider $\bar{x}(t) = x_0$, i.e. no drift. In this case we can solve the equation analytically using Abel's method, as

before:

$$\begin{aligned}\theta(t) &= \frac{1}{\pi} \frac{d}{dt} \left[\int_0^t \frac{a}{\sqrt{t-t'}} dt' \right] = \frac{a}{\pi\sqrt{t}} \\ \Rightarrow \lambda(t) &= \frac{a}{\pi\sqrt{1-t}}\end{aligned}\tag{5.62}$$

and so we see that $\lambda(t)$ contains an (integrable) singularity as $t \rightarrow 1$. We can also treat the case of constant drift where $\bar{x}(t) = x_0 + vt$, such that we are solving:

$$a = \int_0^t \frac{\theta(t')}{\sqrt{t-t'}} e^{-\frac{v^2(t-t')}{4}} dt' \tag{5.63}$$

Setting $\beta(t) = \theta(t)e^{\frac{v^2 t}{4}}$, such that:

$$ae^{\frac{v^2 t}{4}} = \int_0^t \frac{\beta(t')}{\sqrt{t-t'}} dt' \tag{5.64}$$

and again using Abel's method we find that:

$$\theta(t) = \frac{a}{\pi\sqrt{t}} e^{-\frac{v^2 t}{4}} + \frac{av}{\sqrt{4\pi}} \operatorname{erf} \left[\frac{v\sqrt{t}}{2} \right] \tag{5.65}$$

This makes it clear that the singularity will remain when we consider a time-dependent drift as well, which causes a problem as we cannot directly use the same method as we did for finding $j(t)$. However, if we are able to determine the form of (or at least a good approximation) to $\theta(t)$ around $t = 0$ then we can integrate out the first time step which contains the singularity and subtract this off from the rest of the integral, which can then be treated in the exact same way that we used to solve equation 5.23. For the first time step we have the following:

$$a = \int_0^{\Delta t} \frac{\theta(t')}{\sqrt{\Delta t - t'}} e^{-\frac{(\bar{x}(1-t') - \bar{x}(1-\Delta t))^2}{4(\Delta t - t')}} dt' \tag{5.66}$$

Let us make a Taylor expansion to first order of term in the exponential so that we can say $\bar{x}(1-t') \approx \bar{x}(1-\Delta t) + (\Delta t - t')v(1-\Delta t)$. Doing this we are left with:

$$a = \int_0^{\Delta t} \frac{\theta(t')}{\sqrt{\Delta t - t'}} e^{-\frac{v(1-\Delta t)^2(\Delta t - t')}{4}} dt' \tag{5.67}$$

which we have just solved in equation 5.65 (replacing v with $v(1-\Delta t)$). This gives us the approximation of $\theta(t)$ that we need for the first time step, and then for each value

of $t_i > \Delta t$ we can evaluate the first time step as:

$$I_{0,i} = \int_0^{\Delta t} \left[\frac{a}{\pi\sqrt{t'}} e^{-\frac{v^2 t'}{4}} + \frac{av}{\sqrt{4\pi}} \operatorname{erf} \left[\frac{v\sqrt{t'}}{2} \right] \right] \frac{1}{\sqrt{t-t'}} e^{-\frac{(\bar{x}(1-t')-\bar{x}(1-t))^2}{4(t-t')}} dt' \quad (5.68)$$

where $v = v(1 - \Delta t)$. If we make the following definitions: $g_0 = \frac{a}{\pi} e^{-\frac{(\bar{x}(1)-\bar{x}(1-t))^2}{4t}}$ and $g_1 = \frac{a}{\pi} e^{-\frac{u^2 \Delta t}{4}} e^{-\frac{(\bar{x}(1-\Delta t)-\bar{x}(1-t))^2}{4(t-\Delta t)}}$ as well as $f_1 = \frac{au}{\sqrt{4\pi}} \operatorname{erf} \left[\frac{u\sqrt{\Delta t}}{2} \right]$ then we can approximate this integral by replacing the non-singular parts of the integrand by a linear interpolation passing through the values at 0 and Δt :

$$\begin{aligned} I_{0,i} &\approx \int_0^{\Delta t} \left[\frac{(\Delta t - t')g_0 + t'g_1}{\Delta t\sqrt{t'}\sqrt{t-t'}} + \frac{t'f_1}{\Delta t\sqrt{t-t'}} \right] dt' \\ &= \sqrt{\frac{t-\Delta t}{\Delta t}}(g_0 - g_1) + \frac{(2g_0\Delta t + (g_1 - g_0)t)}{\Delta t} \sin^{-1} \left(\sqrt{\frac{\Delta t}{t}} \right) + \frac{2f_1}{3\Delta t} (2t^{3/2} - \sqrt{t-\Delta t}(2t + \Delta t)) \end{aligned} \quad (5.69)$$

This value can then be subtracted off from the total integral at which point we can use exactly the same method as we used to obtain $j(t)$ from equation 5.23.

At this point, given an initial guess for the optimal $\bar{x}(t)$ we are able to obtain numerical approximations for both $j(t)$ and $\lambda(t)$. The final step is to be able to use these new approximations to obtain an updated guess for $\bar{x}(t)$. We do this by using the equation 5.54 to solve for the unknown $\bar{x}(t)$, given the current approximations for $j(t)$ and $\lambda(t)$. This is significantly more difficult for two reasons: firstly it is an integro-differential equation rather than just an integral equation, and secondly because it is nonlinear in the unknown function $\bar{x}(t)$. This means that whatever set of equations we get for $\bar{x}(t)$ at the discrete time points being considered will also be nonlinear, and hence not soluble by simple linear algebra. Nevertheless, we press ahead with essentially the same approach, discretising time in the same way as before but now seeking $I + 1$ nonlinear equations to put in the form $\vec{F}(\vec{x}) = 0$. The first of these is simply that $\bar{x}_0 - x_0 = 0$, imposing the first boundary condition, and the final equation is $\bar{x}_I - \bar{x}_{I-1} = 0$, approximately imposing the second boundary condition $\bar{x}'(1) = 0$. Then for $i = 1, \dots, (I - 1)$ we have:

$$\frac{\bar{x}_{i+1} - 2\bar{x}_i + \bar{x}_{i-1}}{\Delta t^2} + \frac{\lambda_i \bar{x}_i}{4\alpha t_i^{3/2}} e^{-\frac{\bar{x}_i^2}{4t_i}} - \frac{\lambda_i}{4\alpha} \int_0^{t_i} \dots dt' + \frac{j_i}{4\alpha} \int_{t_i}^1 \dots dt' = 0 \quad (5.70)$$

where we have used the simplest possible finite difference approximation for the second derivative of \bar{x} . The final step we need to take is to discretise the integrals. To do this, we take inspiration from the method used to solve the previous two equations. Hence to evaluate:

$$I_1(i) = \int_0^{t_i} \frac{j(t')}{(t_i - t')^{3/2}} (\bar{x}(t_i) - \bar{x}(t')) e^{-\frac{(\bar{x}(t_i) - \bar{x}(t'))^2}{4(t_i - t')}} dt' \quad (5.71)$$

we define $h(t, t') = j(t')(\bar{x}(t) - \bar{x}(t')) e^{-\frac{(\bar{x}(t) - \bar{x}(t'))^2}{4(t - t')}}$. By again replacing the non-singular with a piece-wise linear interpolation of the function we can say that:

$$I_1(i) \approx \sum_{j=1}^{i-1} \int_{t_{j-1}}^{t_j} \frac{(t_j - t')h(t_i, t_{j-1}) + (t' - t_{j-1})h(t_i, t_j)}{\Delta t(t_i - t')^{3/2}} dt' + I_{final} \quad (5.72)$$

These sub-integrals can all be evaluated:

$$\begin{aligned} \int_{t_{j-1}}^{t_j} \frac{(t_j - t')h(t_i, t_{j-1}) + (t' - t_{j-1})h(t_i, t_j)}{\Delta t(t_i - t')^{3/2}} dt' &= \frac{2h(t_i, t_j)(2t_i - t_j - t_{j-1}) - 4h(t_i, t_{j-1})(t - t_j)}{\Delta t \sqrt{t_i - t_j}} \\ &+ \frac{2h(t_i, t_{j-1})(2t_i - t_j - t_{j-1}) - 4h(t_i, t_j)(t - t_{j-1})}{\Delta t \sqrt{t_i - t_{j-1}}} \end{aligned} \quad (5.73)$$

apart from on the final step which we have to consider separately. The problem is that the $(t_i - t')^{3/2}$ doesn't initially look like it is integrable. However, if we make a Taylor expansion of $\bar{x}(t)$ during the final time step, i.e. we say that $\bar{x}(t') \approx \bar{x}(t_i) - (t' - t_i)v(t_i)$ then we find that we are left with:

$$I_{final} \simeq \int_{t_{i-1}}^{t_i} \frac{j(t')v(t_i)}{\sqrt{t_i - t'}} e^{-\frac{(\bar{x}(t_i) - \bar{x}(t'))^2}{4(t_i - t')}} dt' \quad (5.74)$$

Defining $h_2(t, t') = j(t')e^{-\frac{(\bar{x}(t) - \bar{x}(t'))^2}{4(t - t')}}$ we can then replace this non-singular part by the linear interpolation between the values at the grid points to get:

$$I_{final} \approx \frac{2}{3}v(t_i)\sqrt{\Delta t}(2h_2(t_i, t_i) + h_2(t_i, t_{i-1})) \quad (5.75)$$

For the second integral:

$$I_2(i) = \int_{t_i}^1 \frac{\lambda(t)}{(t' - t_i)^{3/2}} (\bar{x}(t') - \bar{x}(t_i)) e^{-\frac{(\bar{x}(t') - \bar{x}(t_i))^2}{4(t' - t_i)}} dt' \quad (5.76)$$

the process is virtually identical except we also have to account for the singularity of λ in the final timestep as well as the singularity at $t' = t_i$. However, as we did with the solution to equation 5.52 we know the form that $\lambda(t)$ should take as $t \rightarrow 1$, and so are able to integrate out the final time step.

At this point we now have $I + 1$ non-linear equations for the $I + 1$ values of $\bar{x}(t)$ at $t = 0, \Delta t, \dots, I\Delta t$, in the form:

$$\begin{aligned} F_0(\vec{x}) &= 0 \\ &\vdots \\ F_I(\vec{x}) &= 0 \end{aligned} \tag{5.77}$$

To solve these we use the Newton-Raphson method[72], which takes an initial guess (which at first is the current guess we have for $\bar{x}(t)$) and iteratively updates it according to:

$$\vec{x} \rightarrow \vec{x}' = \vec{x} - J_F^{-1}(\vec{x}) \vec{F}(\vec{x}) \tag{5.78}$$

where J_F is the Jacobian matrix:

$$\begin{bmatrix} \frac{\partial F_0}{\partial \bar{x}_0} & \cdots & \frac{\partial F_0}{\partial \bar{x}_I} \\ \vdots & \ddots & \vdots \\ \frac{\partial F_I}{\partial \bar{x}_0} & \cdots & \frac{\partial F_I}{\partial \bar{x}_I} \end{bmatrix} \tag{5.79}$$

This is trivial, but laborious to calculate. This procedure is repeated until the equations are solved to within some given tolerance, at which point we have an updated guess for $\bar{x}(t)$ given the current $j(t)$ and $\lambda(t)$. We then repeat the whole procedure again, continuing this until $j(t)$, $\lambda(t)$ and $\bar{x}(t)$ converge to a solution of the three optimal equations (again within some tolerance), at which point we say that $\bar{x}'(t) = v^*(t)$, i.e. the optimal drift velocity.

5.5.1 Performance of numerical solution

Before going on to study in detail the numerical results that we obtain from this approach we pause briefly to run a few tests to verify that this method is giving sensible results. Firstly, given that we have an analytic solution for the case when $v(t)$ is constant, it makes sense to check how close the numerical approximation we get from equation 5.59 is to the exact solution to equation 5.23. Figure 5.5(a) shows an example comparing

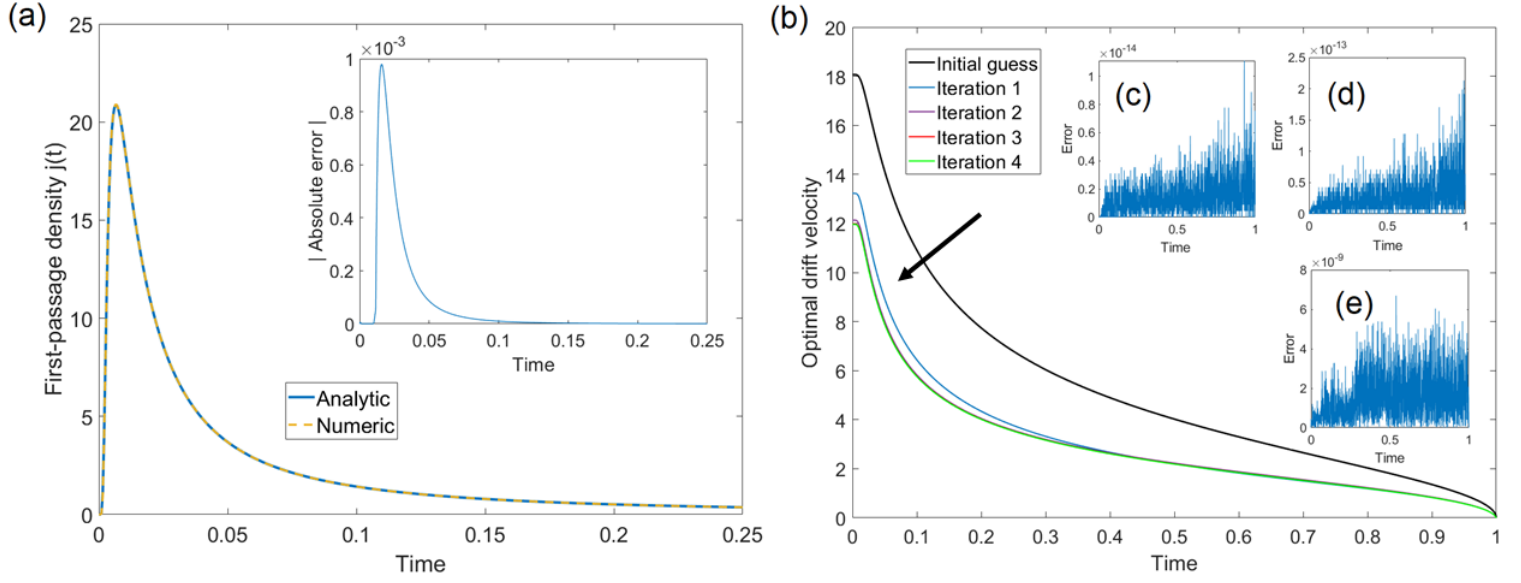


Figure 5.5: (a) As a test of the numerical approach to solve equation 5.23, we look at the case where $v(t) = \text{constant}$ (where we have an exact analytic solution to compare it to). The inset shows the absolute difference between the two curves. Here $x_0 = 0.2$ and $v = 1$. (b) Iterating the numerical procedure to obtain an approximation of the optimal drift velocity. In this example $x_0 = 0.2$ and $\alpha = 0.01$ and the first four iterations after the initial guess are shown (although they are difficult to distinguish as the convergence happens quickly). The initial guess for $v(t)$ is given by equation 5.43. (c)-(e) Absolute difference between the numerical approximations to equations 5.23, 5.52 and 5.54 respectively, evaluated with the converged values of $j(t)$, $\lambda(t)$ and $\bar{x}(t)$.

these when $x_0 = 0.2$, $v = 1$ and where we use 2000 equally spaced intervals to discretise time. Just by eye we see that the agreement is excellent, and the inset plot quantifies the absolute error, i.e. $|j(t) - j_{\text{num}}(t)|$ at each of the times where these functions are calculated. In figure 5.5(b), we then look at how the approximation for the optimal drift velocity converges as we go through iterations of the whole procedure which we have described. Here we are again using $x_0 = 0.2$, and we consider a value of the fitness parameter of $\alpha = 0.01$. The initial guess is given by equation 5.43, and we show $v(t)$ after the first 4 iterations of the numerical procedure. We see that the initial guess converges rapidly to a solution. The inset plots verify that the converged solution actually solves the numerical approximations to the three optimal equations (here after 20 iterations, although it is not necessary to use this many). (c) and (d) are actually not especially informative since they show the (absolute) errors in how close equation 5.59

(and the equivalent for equation 5.52) hold for the final $j(t)$ and $\lambda(t)$. However, these are calculated with the final value of $\bar{x}(t)$, and so really this just tells us how accurate the lower-triangular solver we are using is. (e) is more informative as this shows the absolute error in equation 5.77 *before* we actually update $\bar{x}(t)$ for the last time, given the final values of $j(t)$ and $\lambda(t)$. We see that this error is $O(10^{-9})$, and so it is clear that we have converged on the solution of our three coupled integral equations (or at least to the discrete approximations of these equations).

In general there is no guarantee that this approach will converge from an arbitrary initial condition, and indeed if we provide a particularly poor initial estimate for $v(t)$ it will not. For values of α down to around $\alpha \approx 10^{-3}$ we can get away with using the approximate analytic solution as the initial guess, although eventually the difference becomes too large. However, we can push this to get convergence for even smaller values of α essentially “by induction” — i.e. we use the numerical solution obtained for a slightly higher value of α as the initial guess. The new solution we get can then be used as the initial guess for solving for an even lower value of α , and so on. This works as long as we don’t take the jumps in α to be too large, and by doing this we are able to get a converged solution for values of α down to about $\alpha = 10^{-6}$. With more effort we could probably go even lower, although this didn’t seem necessary here.

5.6 Results

We now report on results that we obtain using this numerical approach to solve for an optimal $v(t)$, choosing a variety of different parameter values. Figure 5.6(a) shows the time dependence of the optimal drift velocity for a fixed value of $x_0 = 0.2$, for various values of α . We see that for large values of α the functional form is almost identical to the approximate analytic result we obtained in equation 5.43, which is reassuring and provides evidence that the numerical approach is working well. However, as α is decreased, we find that there is increasing deviation from this, moving towards a situation where a large initial drift falls off more quickly. The inset plot shows the same behaviour on a log-log scale, and we also see that as α is decreased the time over which the optimal drift goes like $t^{-1/2}$ increases. We postulate that as $\alpha \rightarrow 0$, the form of the time-dependence will converge towards $t^{-1/2}$ over the whole interval. In figure 5.6(b) we look at the initial values of the drift velocity, $v(0)$, again as a function of the fitness parameter α . In the way we’ve formulated the problem this is the primary quantity of interest since it corresponds to the actual decision being made at the present time.

We see that the approximate analytic solution which we derived begins to break down significantly around $\alpha \approx 10^{-2}$, predicting a much larger initial value than it should do. Finally, on the other axis we plot the probability of dying (at the end of the interval, i.e. at $t = 1$) under the numerically obtained optimal drift velocity, again as a function of α . The behaviour we see is fairly intuitive in the sense that as $\alpha \rightarrow 0$, i.e. as energy becomes essentially free, the particle never dies, and as $\alpha \rightarrow \infty$ the probability of dying goes towards the same value as that of a passive particle starting at the same x_0 (given by $\text{erfc}[\frac{x_0}{2}]$). The inset plot shows the same thing but with the probability on a log scale as well, where interestingly we find that $P_d(1) \propto \alpha$ for small values of α . We suspect there is a simple analytic argument which could demonstrate that this is to be expected, but we have not been able to find it.

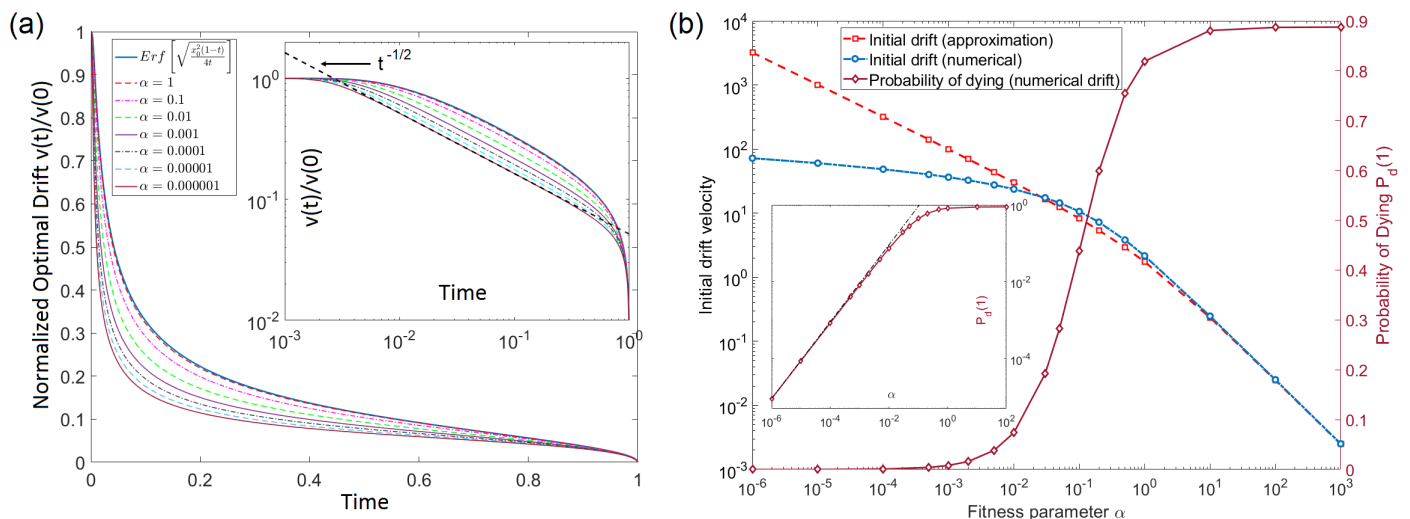


Figure 5.6: (a) Time-dependence of the numerically obtained optimal drift velocity, as a function of α ($x_0 = 0.2$ is fixed in this plot). The time-dependence of the approximate solution (equation 5.43) is also shown for comparison. The inset plot shows the same behaviour on a log-log scale. (b) A comparison of the initial value of the optimal drift, i.e. $v(0)$, as a function of α (again with $x_0 = 0.2$ fixed). We also show how it compares to the approximate analytic solution. On the right hand axis we show the probability of the particle dying under the optimal drift (and the inset shows this on a log scale, where it is clearer that $P_d \propto \alpha$ for small α .)

In figure 5.7(a) we look at the case where we fix $\alpha = 0.05$ and instead consider varying the starting position of the particle. Here we plot the initial value of the optimal drift velocity as well as a best fit to this of the form $v(t) = \frac{c}{x_0}$, where c is a constant pa-

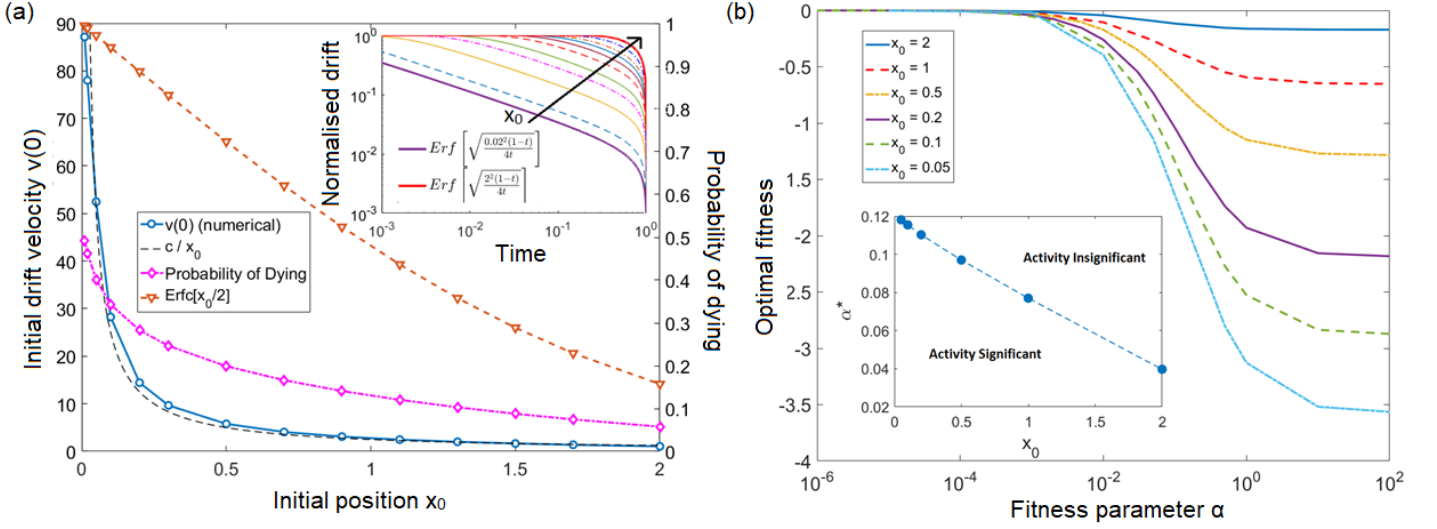


Figure 5.7: (a) (left axis) Initial drift velocity $v(0)$ for fixed $\alpha = 0.05$ as a function of the initial positions. We include a line of best fit of the form $v_0(x_0) = \frac{c}{x_0}$, where c is a parameter we fit. (right axis) The probability of the particle dying under the optimal drift. Also plotted is $\text{erfc}[x_0/2]$, i.e. the probability of a passive particle ($v(t) = 0$) dying. (b) Plots of the fitness functional (equation 5.6) given that the particle moves with the optimal drift velocity, for a variety of α and x_0 values. Inset we plot α^* , defined to be the point of largest curvature in the optimal fitness vs. α curve, and which provides a crude way to partition the phase-space.

parameter. The motivation for this is that if we consider that the time for the root-mean squared displacement due to diffusion to be equal to x_0 , this is given by $\tau \sim x_0^2$. To “match” this, we need an average drift velocity v_{avg} such that $v_{avg}\tau = x_0$, i.e. that $v_{avg} \sim x_0^{-1}$. This gives us a reason to suspect that the initial drift velocity may scale in this way, and indeed we see that fit is very good. This represents an extremely simple heuristic which could potentially be implemented by real microorganisms able to detect the distance to a threat, and which mimics closely the full calculation. The inset plot shows how the time-dependence of the predicted future optimal drift varies with x_0 , on a log-log scale. We see that with smaller x_0 the time-dependence goes like $t^{-1/2}$ for a larger fraction of the interval. The right-hand axes show the probability of dying under the optimal drift, and for comparison we show the value for a passive particle starting at the same initial position. We see that as $x_0 \rightarrow 0$ the probability of a passive particle dying approaches unity quite quickly, but that when we allow the particle to control its drift velocity this is heavily suppressed (presumably this is mainly because we have

included a term of the form $\log(1 - P_d)$ in the fitness, which heavily penalises $P_d \rightarrow 1$). Panel (b) of the figure shows a plot of the actual fitness under the calculated optimal drift velocity, as a function of α , but also for a variety of different values of x_0 . If we identify the value where the derivative of this curve is largest and call it $\alpha^*(x_0)$, then very crudely we can use this to divide the phase-space into two regions; one where the activity plays a significant roll in improving the particle's fitness, and one where the activity is relatively insignificant. This line is plotted as an inset to figure 5.7(b), where we see that as x_0 is increased the region where the activity is significant reduces since the probability of dying becomes small, even with no drift. Overall we see that this division looks quite linear. Speculatively we could make an evolutionary argument which says that a successful organism that can detect a threat at a distance of x should not have a fitness parameter α that falls too far above this line.

Overall in this chapter we have studied a relatively simple continuous decision making process. We have loosely followed the basic structure for FSM, in which decisions are made by modelling a large number of hypothetical trajectories. However rather than explicitly enumerating future trajectories we have instead calculated how the probability distribution of the particle evolves under an arbitrary set of decisions - and then optimised these afterwards to maximise a particular fitness functional that we argue gives a rough estimate of the number of paths available to the particle. As such, whether we can really consider this to be an authentic application of FSM to a fully continuous system is somewhat questionable. Nevertheless, it remains an interesting problem to study in its own right. It's quite possible that the most significant contribution in this chapter could turn out to be the method which we used to derive an integral equation for $j(t)$, the first-passage probability density, when we have an arbitrary drift velocity $v(t)$ (or an arbitrary moving absorbing boundary), as well as the subsequent approximation to P_d in the limit that $v(t)$ is small enough. These kind of first passage problems appear in many different contexts, and to the best of our knowledge the approach we have taken here makes a novel contribution to the literature. For example, geometric Brownian motion[75] is used in the mathematical finance literature with the Black-Scholes model to model the pricing of options and often involves solving similar first-passage time problems[76], therefore it is possible our work could find a use in that area.

Chapter 6

Conclusions and outlook

Throughout this thesis we have concerned ourselves with thinking about ways in which the principle of future state maximisation can be applied to decision making in a variety of different contexts. Whilst the basic principle is extremely general, and intuitively we may have a sensible idea of what it means for an agent to “maximise its options”, in practice applying FSM to complex systems is not trivial. As we have seen, there is a serious issue with the most straightforward approaches to applying FSM in terms of the computational resources they require — since making a single decision requires the modelling of a large number of hypothetical future trajectories. Furthermore, as the planning time horizon increases the number of potential future states generally increases exponentially in that horizon time, making the direct application of FSM over large time intervals essentially impossible. There are also questions about how best to quantify the degeneracy in potential future states. Whilst the “empowerment” framework[5] provides a rigorous formulation in the language of information theory, it (a) is an extremely difficult quantity to calculate and (b) can lead to some rather unintuitive results. For example, consider the definition of the discrete visual states given in chapter 4. Let us first imagine an (unrealistic) situation where there are only two possible visual states available at some future time, $[0, 1, 0, 0, 0, 0, 0, 0]$ and $[0, 0, 1, 0, 0, 0, 0, 0]$. Since these are distinct, the empowerment is $\log(2)$ — however we can see that intuitively these states are extremely similar. If we imagine now the case where the two states are instead $[1, 0, 0, 0, 0, 0, 0, 0]$ and $[0, 0, 0, 1, 1, 1, 1, 1]$, the empowerment is still $\log(2)$, however it is clear that these two states differ significantly. It seems that in a situation where it is possible to define a metric comparing how similar any two given states are then this should be incorporated into whatever measure we use to quantify the degeneracy of future states.

In Chapter 3 we looked at potential ways to address some of these issues. Firstly, we looked at how if we limit ourselves to studying discrete systems with a state space which can be described by a graph then there is a simple and efficient way of counting the number of paths available to an agent involving taking powers of the system’s adjacency matrix. This allowed us to extend the time horizon τ to significantly larger values than we could by explicit enumeration of future trajectories, and as a demonstration of this we applied it to solving a maze. Unfortunately, in its current form, this approach can only really be applied to “toy problems”, since the state space for more complex systems rapidly becomes too large to handle in this way. We then considered an alternative way to measure the degeneracy of accessible future states in terms of a function which quantifies how different any two given states are — addressing the issue we raised with empowerment at the end of the previous paragraph. However, the approach we take does not have a general, strong theoretical foundation, and it would be nice if it could be combined with the empowerment framework more explicitly. Finally, we applied this to a simple game where we were able to effectively extend the time horizon considered quite significantly by supplementing FSM over a short time horizon with a random search over a longer time horizon.

Chapter 4 represented our primary application of FSM to a more complicated system. Here we considered a group of agents equipped with simple visual sensors which detect a projection of the world around them. We found that by each agent individually moving so as to maximise the amount of control it has over its future visual environment, based on some assumptions about how the other agents will move, that this induces highly ordered, robust and cohesive collective motion. This approach differs significantly from other common models of collective motion in that it starts from a simple, low-level motivational principle and everything else emerges spontaneously. As such, it can be viewed as FSM being used to explain *why* agents favour moving collectively, rather than just how they do so. We also demonstrated that the behaviour which emerges from applying FSM explicitly can be closely mimicked by a neural network which only has access to the presently available visual information, i.e. without the need to actually model future paths. This shows that, at least in this example, that it is possible for the features which arise from the complicated considerations of FSM to be replicated with a simple heuristic, and it seems plausible that this could be true for a much wider range of scenarios as well. This ability for the results of FSM to be closely replicated by significantly simpler heuristics is absolutely necessary if it going to be used to explain behaviour observed in natural systems, since it is not reasonable to expect animals to do anything similar to

these kind of calculations themselves. As we went on to look at larger groups of agents, the spontaneous dynamics which emerged in the continuous version of the model were remarkably rich — especially considering that we restricted ourselves to working in 2D. A natural extension of this work would be to consider the 3D case, where we might expect even more interesting dynamics to emerge. Lastly in this chapter we considered applying FSM at two vastly different scales — one to control the swarm’s centre of mass trajectory over a large time horizon and the other over a much shorter time horizon to motivate the individual decisions. As an example, we showed how this could be used to guide a swarm out of a simple maze.

Finally, in Chapter 5 we studied a completely continuous decision making process, where we took time as well as the state/action space available to the agent to be continuous. Here, a diffusing particle with control over its drift velocity had to choose how to move so as to avoid a stationary threat. The approach we took differed from any of the previous chapters in that rather than explicitly enumerating future paths, we instead considered optimising a fitness functional, arguing that this loosely mimics FSM. We derived an exact integral equation to solve for the first-passage to the threat probability density for arbitrary, time-dependent drift velocities, and which we noted could be generalised to situations with absorbing boundaries which moved as well. We then went on to obtain an approximate analytic solution for the optimal drift as well as developing a numerical method to solve for this exactly, again finding that relatively simple heuristics emerge that are able to produce similar behaviour.

6.1 Ideas for future work

To fully frame FSM as a viable procedure for generating the behaviour of complex, artificial agents there is still much work to do. Being able to prune the future state space in such a way so as to be able to explore it more efficiently provides a real challenge, and we need to be able to give a way of either recognising or learning to recognise future avenues which are irrelevant or extremely implausible — and so not necessary to enumerate. For example, if a human is reasoning about the potential future options they have available to them they do not have to consider every possible sequence of micro-actions that they can take, and indeed even if they wanted to, they are not able to accurately model all of these details anyway. Nevertheless, we are reasonably good at imagining a set of extremely coarse-grained, but plausible, future trajectories that we could follow, and these generally become more accurate as we increase our knowledge

about the environment we're in. Whilst getting anywhere close to this seems like it would require a number of different advances there are certain ideas which could be explored which might be able to steer us in this kind of direction. We discuss some of these below.

One possibility might be to combine FSM with the “options” framework[77], introduced in the context of reinforcement learning to combat a similar issue in which one does not want to explore using only long sequences of primitive actions. The idea here is to generalise primitive actions to include temporally extended courses of actions. As such, an option $\langle I, \pi, \beta \rangle$ consists of a policy π which provides a probability of taking any primitive action from any given state *whilst the option is activated*, a termination condition β , which gives the probability of the option terminating in any given state and an input set I which is the subset of states from which the option can be initiated from. Essentially, an option is a policy which tells the agent how to act until it stochastically terminates according to β , when a different option can be chosen. We can think of this as a generalisation of a “macro-action”, which is simply a sequence of primitive actions combined together. Rather than exploring future states in which the branches correspond to different primitive action choices, the idea here would be to instead consider branches corresponding to choices of different options. The question then would be, “how should these options be implemented?” In some situations it might be natural to design them by hand, however we would like a more general approach in which the options can be learned. One possibility could be to use an evolutionary algorithm[78], with a fitness based on evaluating the number of possible states that an agent is able to access. The agents could have a fixed number of potential options they can activate (each encoded by a neural network, say), which can be adjusted via evolution in order to alter how they operate. If the resultant agents can be evaluated in some way that gives an accurate representation of the amount of potential states that they are able to access, then, naturally, this process should lead to options being developed which give the agent access to the widest range of possible future states. We could expect that this would spontaneously lead to the agent learning about structure in the environment, as well as developing options which are “reusable”, i.e. useful in a number of different scenarios that the agent may encounter. The hope would be that these things would develop much more rapidly than if we simply used a fitness based on the agent's survival alone. We could also imagine co-evolving a control or “master” policy which selects options in a given state. This would then be very analogous to hierarchical reinforcement learning[79].

A significant obstacle to this approach is how to actually measure the number of states

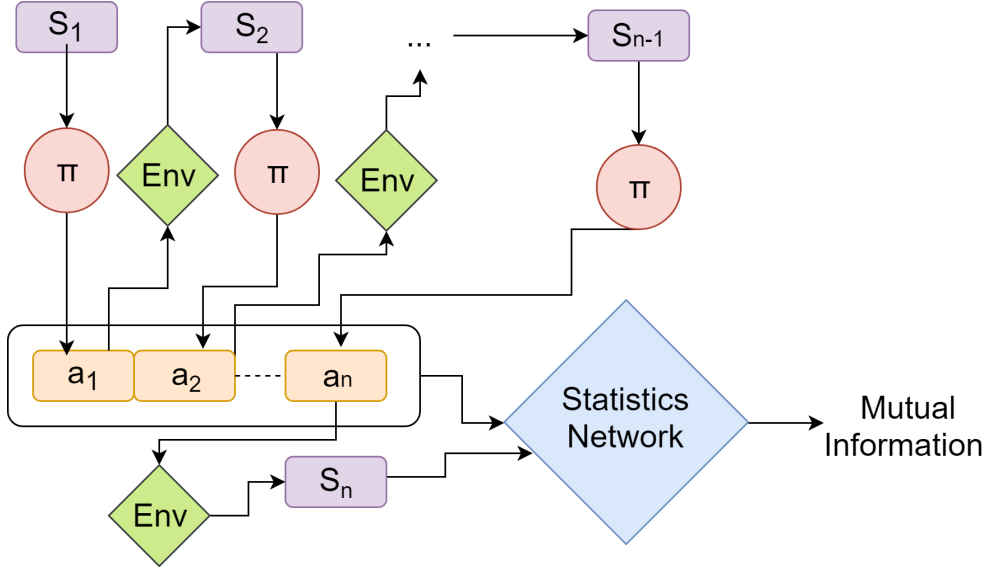


Figure 6.1: A sketch of how we might calculate the empowerment of systems with high-dimensional state/action spaces. Here, a policy (i.e. a neural network) takes in the state of the system S_i and outputs an action a_i . This action is then passed into the environment (represented by the green diamond) to produce a new state. The statistics network (also a neural network) takes a batch of action sequences and final states and estimates the mutual information, using the technique described in [80]. Since the policy is a differentiable function, it should be possible to use stochastic gradient descent in order to optimise π so as to maximise the mutual information between action sequences and final states, i.e. to calculate the agent’s empowerment.

available to an agent. Obviously this has been a key topic of this thesis, however it’s likely that none of the methods discussed so far would be particularly suitable here. One crude approach we could take would be to keep a memory bank of states that the agent has seen. If a state we encounter is “more different” than some amount ϵ , given by some difference measure we have $d(\mathbf{x}_1, \mathbf{x}_2)$ (as in section 3.2), then we add the state to the memory bank and it contributes to a count. If we imagine initialising an agent many times in many different realisations of the environment, we could build up a memory of the states it has been able to reach with its current options/master policy, and use this to calculate its fitness. It is hard to say how well this approach would work, nevertheless running some experiments along these lines could be very interesting.

Another idea is to use the recently proposed method called “Mutual Information Neural Estimation” [80], which allows for the mutual information between high dimensional

variables to be calculated in a much more efficient manner than was previously possible. It uses the fact that it is possible to write the mutual information between two random variables X and Y , with values in \mathcal{X} and \mathcal{Y} respectively, in the following way:

$$I(X; Y) = \sup_T \mathbb{E}_{\mathcal{P}_{X,Y}} [T(X, Y)] - \log \left(\mathbb{E}_{\mathcal{P}_X \otimes \mathcal{P}_Y} \left[e^{T(X,Y)} \right] \right) \quad (6.1)$$

where T is in the set of functions $T : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ that have finite values over the expectations. The expectation $\mathbb{E}_{\mathcal{P}_{X,Y}}$ means that values of X and Y are sampled together from the joint distribution, whereas $\mathbb{E}_{\mathcal{P}_X \otimes \mathcal{P}_Y}$ refers to an expectation where each variable is sampled with respect to its marginal distribution instead. The crucial insight is that if you choose $T = T_\theta$ to be an expressive function capable of some generalisation between states, like a deep neural network (with weights θ), then you can calculate an estimate for the mutual information by simply maximising equation 6.1 using stochastic gradient descent on batches of samples of X and Y . The authors refer to T_θ as the “statistics network”. This approach might allow for an accurate approximation to an agent’s empowerment to be calculated for much higher dimensional state/action spaces, since empowerment is defined in terms of maximising the mutual information between action sequences and future states, $I(A_t^n; S_{t+n})$, as described in Chapter 2. A basic illustration of how this could work is sketched out in figure 6.1. If we take a policy π_ϕ , itself a neural network with weights ϕ , which maps every state to the probability of taking each available action, then we can generate batches of action sequences and final states to train the statistics network on. Of course this would just give the mutual information between action sequences and final states under the *current policy*, which will not necessarily correspond to the empowerment. However, since the policy π is itself a differentiable function, we can propagate the gradients back through the statistics network to alter the policy, so as to maximise the mutual information. If this works, it would then give us the empowerment-maximising policy, which could then either be used directly to explore or make decisions, or could be used to estimate the agent’s empowerment in any given state.

Finally, another interesting area to explore would be how the approaches we have considered in this thesis need to be modified when we have some kind of variable state desirability. Although FSM is most naturally applicable to situations where the agent isn’t trying to achieve a specific goal it is nevertheless easy to imagine situations in which the agent would still have some prior knowledge about certain types of states being preferable to others. In this case, rather than purely maximising over the number/density of available potential future states the agent would want to give a different

weighting to different states. In the simplest case where we are counting “distinct” states this could be easily achieved — for example we could just define a weighting function $\omega(s)$ that provides a weight to each state s and then calculate a weighted sum over distinct states: $\sum_{\text{states}, i} \omega(s_i) \mathbb{1}(s_i \text{ is distinct})$. This could also easily be used with the average difference measure discussed in Chapter 3 (and used in the continuous version of the model in Chapter 4) by calculating: $\sum_{\text{states}, i} \omega(s_i) \frac{1}{N_{\text{states}}} \sum_{\text{states}, j} d(s_i, s_j)$, where d is the difference measure between states we are using. However, it is not trivial to see the obvious ways in which the empowerment and causal entropic forces frameworks would have to be modified to take this weighting into account.

Appendix A

List of supplementary movies

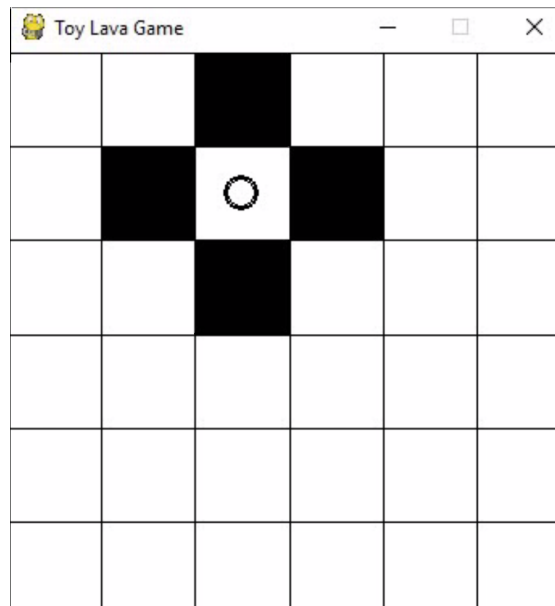


Figure A.1: SI Movie 1: Examples of some of the successes (and failures) of the method described in section 3.2.2 applied to a simple game. Here the agent has to plan over a large number of time steps in order to build itself a shelter in order to survive.

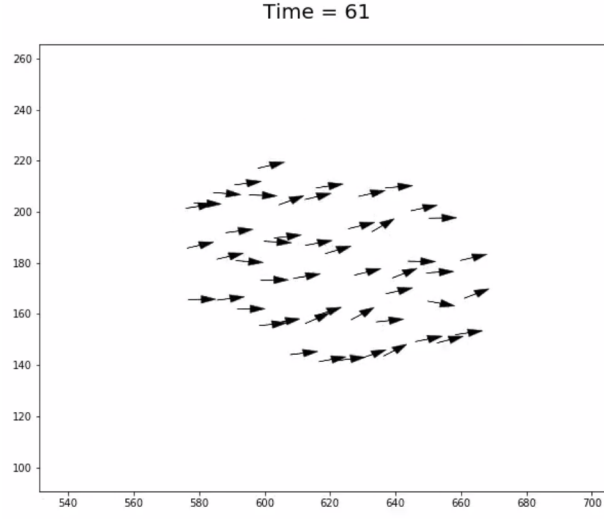


Figure A.2: SI Movie 2: video showing an example of a swarm which spontaneously emerges using the FSM approach in Chapter 4. Here we use the nominal set of parameters, $N = 50$, $\tau = 4$, $n_s = 40$, $v_0 = 10$, $R = 1$, $\Delta\theta = 15^\circ$ and $\Delta v = 2$.

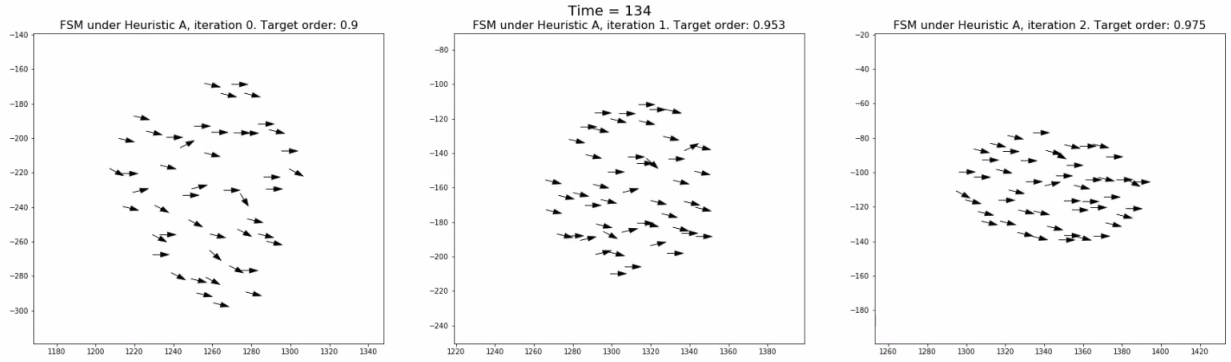


Figure A.3: SI Movie 3: Convergence to a more self-consistent heuristic. Here we show side-by-side the first three iterations using the “collective target order heuristic”, described in section 4.3.

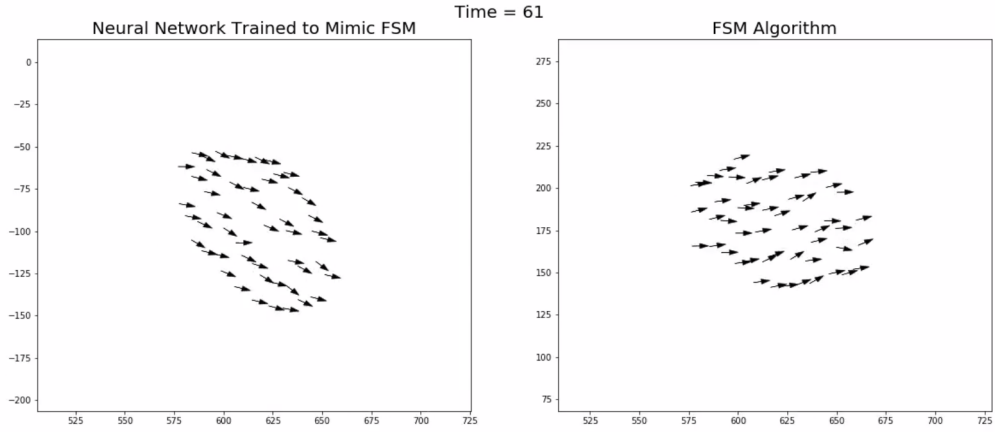


Figure A.4: SI Movie 4: A side-by-side comparison of the full FSM algorithm (right, as in SI Movie 1) with the results of running the neural network which was trained to mimic it (left), as described in section 4.4.

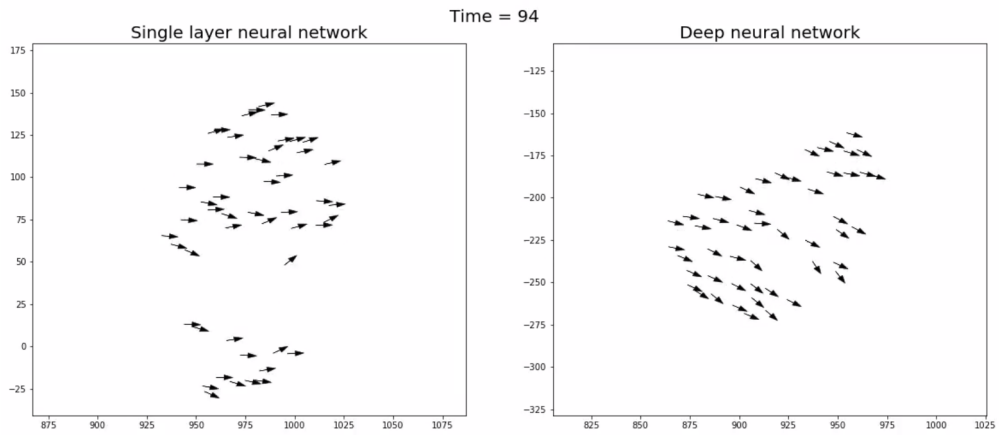


Figure A.5: SI Movie 5: A side-by-side comparison of training a heuristic using a single-layered neural network vs. using a multi-layered (deep) neural network.

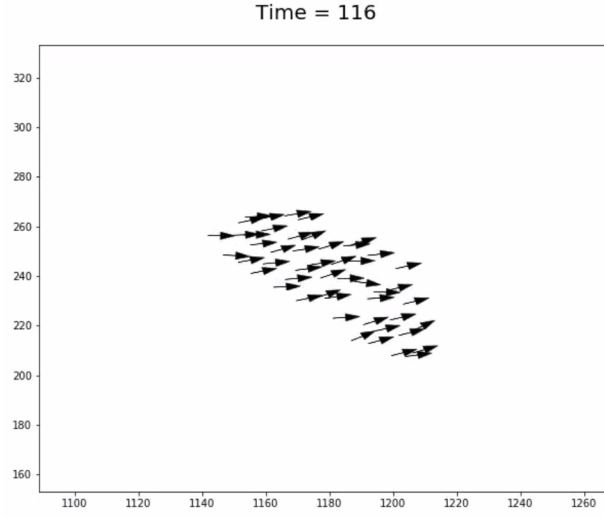


Figure A.6: SI Movie 6: Results of running the continuous visual state version of the model, described in section 4.8. The parameters used here are $N = 50$, $\tau = 5$, $v_0 = 10$, $R = 1$, $\Delta\theta = 15^\circ$ and $\Delta v = 2$.

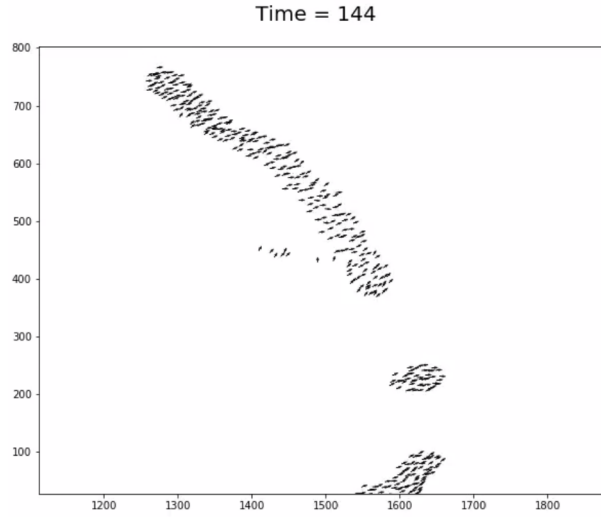


Figure A.7: SI Movie 7: Results of running the original (discrete visual sensor) version of the model, but with $N = 500$ and $\tau = 6$.

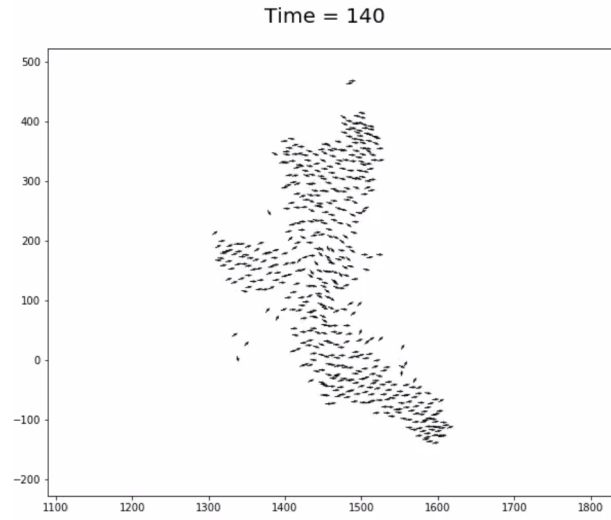


Figure A.8: SI Movie 8: Results of running the continuous visual sensor version of the model, with $N = 500$ and $\tau = 5$.

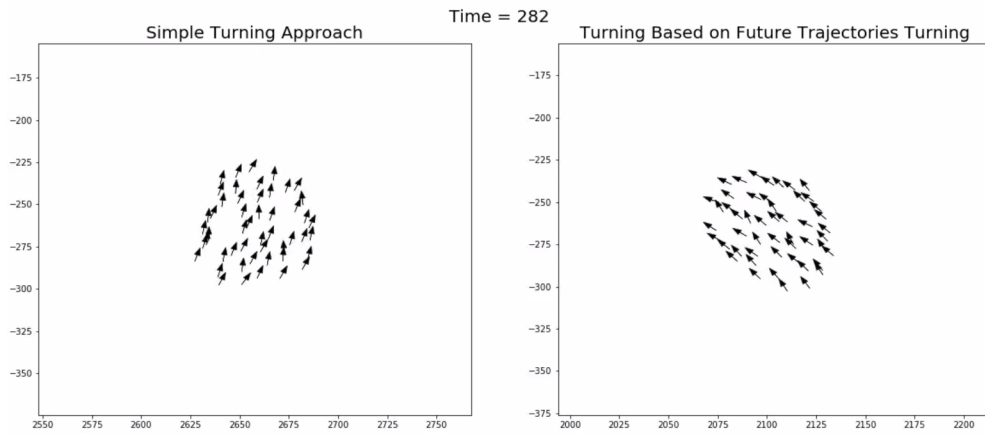


Figure A.9: SI Movie 9: A side-by-side comparison of the two approaches of making the swarm turn, described in section 4.9.

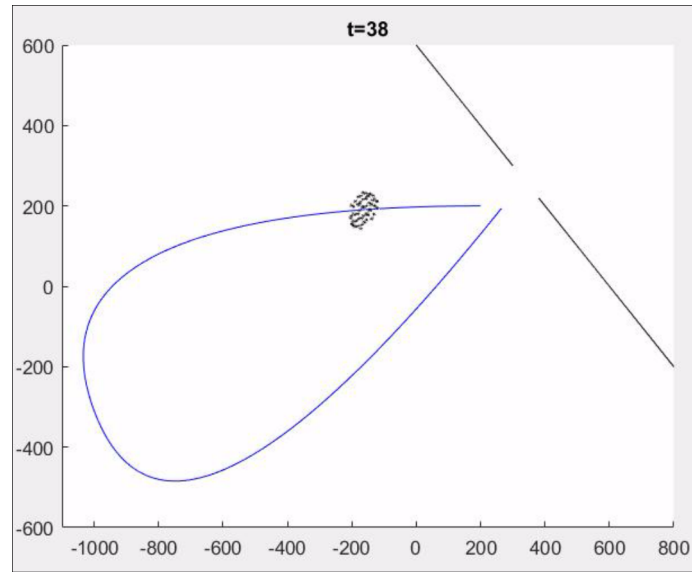


Figure A.10: SI Movie 10: Example of a swarm following a specified trajectory, using the approach described in section 4.9.2.

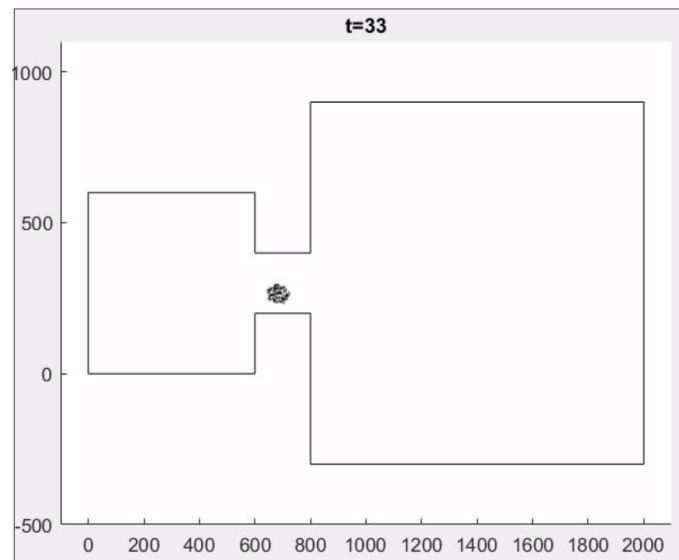


Figure A.11: SI Movie 11: Example of applying FSM at two different scales - at the level of the swarm's centre of mass trajectory as well as motivating the individual decisions. Described in section 4.9.3.

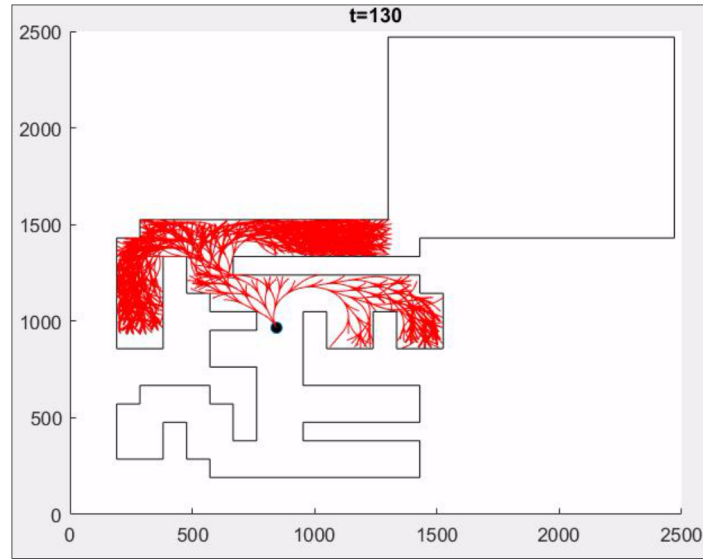


Figure A.12: SI Movie 12: Visualisation of the future tree search of centre of mass trajectories for a swarm exploring a maze. As described in section 4.9.3.

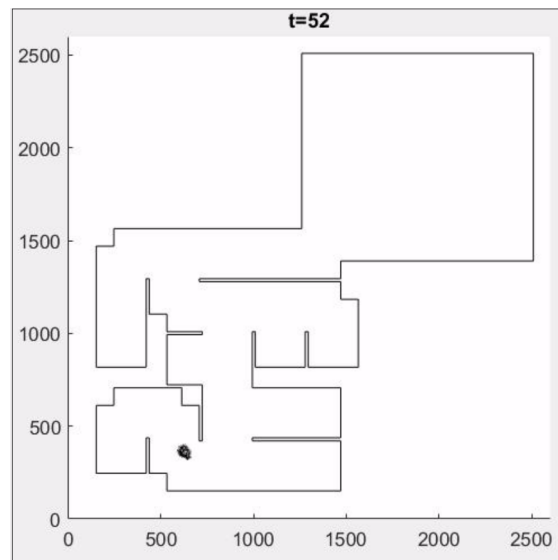


Figure A.13: SI Movie 13: Example of a swarm following the centre of mass trajectory, generated via FSM as described in section 4.9.3.

Bibliography

- [1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.
- [2] Kun-Hsing Yu, Ce Zhang, Gerald Berry, Russ Altman, Christopher Ré, and Daniel Rubin. Predicting non-small cell lung cancer prognosis by fully automated microscopic pathology image features. *Nature Communications*, 7:12474, 2016.
- [3] David Silver et al. Mastering the game of go without human knowledge. *Nature*, 550:354–359, 2017.
- [4] Richard Ryan and Edward Deci. Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary Educational Psychology*, 25:54–67, 2000.
- [5] Alexander Kluybin, Daniel Polani, and Chrystopher Nehaniv. Empowerment: A universal agent-centric measure of control. In *The 2005 IEEE Congress on Evolutionary Computation vol. 1*, pages 128–135, 2005.
- [6] Alexander Wissner-Gross and Cameron Freer. Causal entropic forces. *Physical Review Letters*, 110:168702, 2013.
- [7] Gianluca Baldassarre and Marco Mirolli. *Intrinsically Motivated Learning in Natural and Artificial Systems*. Springer, 2013.
- [8] Andrew Barto, Satinder Singh, and Nuttapon Chentanez. Intrinsically motivated learning of hierarchical collections of skills. In *International Conference on Developmental Learning*, 2004.
- [9] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016.

- [10] Joshua Achiam and Shankar Sastry. Surprise-based intrinsic motivation for deep reinforcement learning. *arXiv preprint arXiv:1703.01732*, 2017.
- [11] Clark Hull. *Principles of Behavior - An introduction to Behavior Theory*. New York: Appleton-Century-Crofts, 1943.
- [12] Ivan Pavlov. *Conditional Reflexes*. New York: Dover Publications, 1927.
- [13] Harry Harlow. Learning and satiation of response in intrinsically motivated complex puzzle performance by monkeys. *Journal of Comparative and Physiological Psychology*, 43:289–294, 1950.
- [14] Kay Montgomery. The role of exploratory drive in learning. *Journal of Comparative and Physiological Psychology*, 47:60–64, 1954.
- [15] A Klopf. *The Hedonistic Neuron: A Theory of Memory, Learning and Intelligence*. Hemisphere, Washington, 1982.
- [16] Robert White. Motivation reconsidered: The concept of competence. *Psychology Review*, 66:297–333, 1959.
- [17] Juergen Schmidhuber. Artificial curiosity based on discovering novel algorithmic predictability through coevolution. In *IEEE Congress on Evolutionary Computation*, volume 3, pages 1612–1618, 1999.
- [18] Daniel Berlyne. A theory of human curiosity. *British Journal of Psychology*, 45:180–191, 1954.
- [19] Alexander Klyubin, Daniel Polani, and Chrystopher Nehaniv. All else being equal be empowered. In *Advances in Artificial Life, European Conference on Artificial Life*. Springer, 2005.
- [20] Claude Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 1948.
- [21] Richard Blahut. Computation of channel capacity and rate-distortion functions. In *IEEE Transactions on Information Theory*, volume 18, pages 460–473, 1972.
- [22] Suguru Arimoto. An algorithm for computing the capacity of arbitrary discrete memoryless channels. In *IEEE Transactions on Information Theory*, volume 18, pages 14–20, 1972.

- [23] Mikhail Protenko. *Guided Self-Organization: Inception*. Springer, 2014.
- [24] Alexander Kluybin, Daniel Polani, and Chrystopher Nehaniv. Keep your options open: an information-based driving principle for sensorimotor systems. *PloS ONE*, 3, 2008.
- [25] Tom Anthony, Daniel Polani, and Chrystopher Nehaniv. Impoverished empowerment: 'meaningful' action sequence generation through bandwidth limitation. In *European conference on artificial life*, 2009.
- [26] Christopher Salge, Christian Guckelsberger, Rodrigo Canaan, and Tobias Mahlmann. Accelerating empowerment computation with uct tree search. *arXiv preprint arXiv:1803.09866*, 2018.
- [27] Shakir Mohamed and Danilo Jimenez Rezende. Variational information maximisation for intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pages 2125–2133, 2015.
- [28] Tobias Jung, Daniel Polani, and Peter Stone. Empowerment for continuous agent-environment systems. *Adaptive Behavior*, 19, 2011.
- [29] Philippe Capdepuy, Daniel Polani, and Chrystopher Nehaniv. Maximization of potential information flow as a universal utility for collective behaviour. In *IEEE Symposium on Artificial Life*, pages 207–213, 2007.
- [30] Phillipe Capdepuy. *Principles of Perception-Action Loops and Collective Behaviours*. PhD thesis, University of Hertfordshire, 2010.
- [31] Phillipe Capdepuy, Daniel Polani, and Chrystopher Nehaniv. Perception-action loops of multiple agents: informational aspects and the impact of coordination. *Theory in Biosciences*, 131:149–159, 2012.
- [32] Tamas Vicsek and Anna Zafeiris. Collective motion. *Physics Reports*, 512:71–140, 2012.
- [33] James Shapiro and Martin Dworkin. *Bacteria as Multicellular Organisms*. Oxford University Press, 1997.
- [34] Madeleine Beekman, David Sumpter, and Francis Ratnieks. Phase transition between disordered and ordered foraging in pharaoh's ants. *Proceedings of the National Academy of Sciences*, 98:9703–9706, 2001.

- [35] Jerome Buhl et al. From disorder to order in marching locusts. *Science*, 312:1402–1406, 2006.
- [36] Christophe Becco, Nicolas Vandewalle, Johann Delcourt, and Pascal Poncin. Experimental evidences of a structural and dynamic transition in fish school. *Physica A: Statistical Mechanics and its Applications*, 367:487–493, 2006.
- [37] Andrea Cavagna et al. Scale-free correlations in starling flocks. *Proceedings of the National Academy of Sciences*, 107:11865–11870, 2010.
- [38] Michele Ballerini et al. Interaction ruling animal collective behavior depends on topological rather than metric distance: Evidence from a field study. *Proceedings of the National Academy of Sciences*, 105:1232–1237, 2008.
- [39] Daniel Pearce. *Swarming*. PhD thesis, University of Warwick, 2014.
- [40] Ying Tan and Zhong-yang Zheng. Research advances in swarm robotics. *Defence Technology*, 9:18–39, 2013.
- [41] Craig Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21:25–34.
- [42] Hugues Chaté, Francesco Ginelli, Guillaume Grégoire, and Franck Raynaud. Collective motion of self-propelled particles interacting without cohesion. *Physical Review E*, 77:046113.
- [43] Hugues Chaté and Francesco Ginelli. Relevance of metric-free interactions in flocking phenomena. *Physical Review Letters*, 105:168103, 2010.
- [44] Atsuyuki Okabe, Barry Boots, Kokichi Sugihara, and Sung Nok Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams, 2nd Edition*. Wiley, 2000.
- [45] Daniel Pearce and Matthew Turner. Density regulation in strictly metric-free swarms. *New Journal of Physics*, 16:082002, 2014.
- [46] Daniel Pearce, Adam Miller, George Rowlands, and Matthew Turner. Role of projection in the control of bird flocks. *Proceedings of the National Academy of Sciences*, 111:10422–10426.

- [47] David Wilson. Generating random spanning trees more quickly than the cover time. In *Proceedings of the twenty-eight annual ACM Symposium on Theory of computing*, pages 296–303, 1996.
- [48] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [49] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [50] François Chollet. Keras. <https://github.com/keras-team/keras>, 2018.
- [51] J Hopcroft and R Tarjan. Algorithm 447: Efficient algorithms for graph manipulation. *Communications of the ACM*, 16:372–378, 1973.
- [52] M de Berg, M van Kreveld, M Overmars, and O Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer, 2000.
- [53] Thierry Mora and William Bialek. Are biological systems poised at criticality? *Journal of Statistical Physics*, 144:268–302, 2011.
- [54] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Second Edition*. Springer, 2009.
- [55] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [56] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- [57] Shannon R. Butler, Elizabeth C. Hosinski, Jeffrey Lucas, and Esteban Fernandez-Juricic. Social birds copy each other’s lateral scans while monitoring group mates with low-acuity vision. 121:21–31, 11 2016.
- [58] Peter Pacheco. *Parallel Programming with MPI*. Morgan Kaufmann, 1996.
- [59] Hafsa Deddi, Hazel Everett, and Sylvain Lazard. Interpolation with curvature constraints. In *Proceedings of the International Conference on Curves and Surfaces*, volume 2, 1999.
- [60] Stewart Harris. *An Introduction to the Theory of the Boltzmann Equation*. Dover Books on Physics, 2011.

- [61] John Toner and Yuhai Tu. Flocks, herds, and schools: A quantitative theory of flocking. *Phys. Rev. E*, 58:4828–4858, Oct 1998.
- [62] Sidney Redner. *A Guide to First-Passage Processes*. Cambridge University Press, 2001.
- [63] Hannes Risken and Till Frank. *The Fokker-Planck Equation: Methods of Solution and Applications*. Springer, 1996.
- [64] Ronald N. Bracewell. *The Fourier Transform and Its Applications*. McGraw-Hill Series in Electrical and Computer Engineering, 1999.
- [65] Rudolf Gorenflo and Sergio Vessella. *Abel Integral Equations: Analysis and Applications*. Springer, 1991.
- [66] Richard Weiss. Product integration for the generalized abel equation. *Mathematics of Computation*, 26(117), 1972.
- [67] H. J. Sussmann and J. C. Willems. 300 years of optimal control: from the brachystochrone to the maximum principle. *IEEE Control Systems Magazine*, 17(3):32–44, June 1997.
- [68] Dominic Jordan and Peter Smith. *Nonlinear Ordinary Differential Equations: An Introduction for Scientists and Engineers. Fourth Edition*. Oxford University Press, 2007.
- [69] Jichao Zhao and Robert M. Corless. Compact finite difference method for integro-differential equations. *Appl. Math. Comput.*, 177(1):271–288, 2006.
- [70] Jafar Saberi-Nadijafi and Mohamadreza Tamamgar. The variational iteration method: A highly promising method for solving the system of integro-differential equations. *Computers and Mathematics with Applications*, 56:346–351, 2008.
- [71] Edyta Hetmaniok, Damian Słota, Tomasz Trawiński, and Roman Witula. Usage of the homotopy analysis method for solving the nonlinear and linear integral equations of the second kind. *Numerical Algorithms*, 67(1):163–185, 2014.
- [72] C. Kelley. *Solving Nonlinear Equations with Newton’s Method*. Society for Industrial and Applied Mathematics, 2003.
- [73] Eric Weisstein. Simpson’s rule. <https://mathworld.wolfram.com/SimpsonsRule.html>, 2018.

- [74] William Press, Saul Teukolsky, William Vetterling, and Brian Flannery. *Numerical Recipes in C++: The Art of Scientific Computing*. Cambridge University Press, 2002.
- [75] Sheldon Ross. *Introduction to Probability Models, 11th Edition*. Elsevier, 2014.
- [76] Davide Valenti, Bernardo Spagnolo, and Giovanni Bonanno. Hitting time distributions in financial markets. *Physica A*, 382:311–320, 2007.
- [77] Richard Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112:181–211, 1999.
- [78] Thomas Weise. *Global optimization algorithms – theory and application*, 2008.
- [79] Andrew Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems: Theory and Applications*, 13:41–77, 2003.
- [80] Mohamed Belghazi, Aristide Baratin, Sai Rajeswar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and R Devon Hjelm. Mutual information neural estimation. *arXiv preprint arXiv:1801.04062*, 2018.